5 Fundamentals of data representation

Updated 15th January 2019

5.1 Number systems

Chapter 5.1.1 Natural numbers

|  |  |
| --- | --- |
| T1 | Babylonian 1.svg  Babylonian 10.svg |
| 1a | 7 |
| 1b | 77 |
| 1c | 1996 |
| 2a | 1275 |
| 2b | 20100 |
| 3 | 3 **possible answers:**  **½n (1 + n)**  **½n2 + ½n**  **n(1 + n)/2** |
| P1 | **Python 3 answer**  **n = int(input("Enter a value for n: "))**  **Answer = 0**  **for Count in range(1, n + 1):**  **Answer += Count**  **print("The sum of the first n natural numbers is", Answer)** |
| P2 | **Python 3 answer**  **n = int(input("Enter a value for n: "))**  **print("The sum of the first n natural numbers is", int(0.5\*n\*\*2+0.5\*n))** |

Chapter 5.1.2 Integer numbers

|  |  |
| --- | --- |
| 1a | **Yes** |
| 1b | **No** |
| 1c | **No** |
| 2 | **Yes,** (367 x 42)/7 = 367 x 6 = 2202 |
| P1 | **In VB.Net there are several different data types that can be used to store integer values.**   |  |  | | --- | --- | | **Data type** | **Range** | | **Byte** | 0 to 255 | | **Short** | -32,768 to 32,767 | | **UShort** | 0 to 65535 | | **Integer** | -2,147,483,648 to 2,147,483,647 | | **UInteger** | 0 to 4,294,967,295 | | **Long** | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 | | **ULong** | 0 to 18,446,744,073,709,551,615 | |

Chapter 5.1.3 Rational numbers

|  |  |
| --- | --- |
| 1a | False |
| 1b | True |
| 1c | True |
| P1 | **VB.Net**  Module Module1  Sub Main()  Dim x As Integer  Dim y As Integer  Dim q As Integer  Dim r As Integer  Console.Write("Enter a value for x: ")  x = Console.ReadLine  Console.Write("Enter a value for y: ")  y = Console.ReadLine  GetQAndR(x, y, q, r)  Console.WriteLine("The value of q is: " & q)  Console.WriteLine("The value of r is: " & r)  Console.ReadLine()  End Sub  Sub GetQAndR(ByVal x As Integer, ByVal y As Integer, ByRef q As Integer, ByRef r As Integer)  Debug.Assert(x >= 0)  Debug.Assert(y > 0)  r = x  q = 0  While r >= y  r = r - y  q = q + 1  End While  Debug.Assert(x = y \* q + r)  Debug.Assert(0 <= r)  Debug.Assert(r < y)  End Sub  End Module |
| 2a | 5/125 = 1/25 = 1/5 \* 1/5, so yes |
| 2b | 16/1024 = 1/64 = ½ \* ½ \* ½ \* ½ \* ½ \* ½, so yes |
| 2c | 2/3, prime factor of denominator is 3 – so no |
| 3a | 5.3 need to place a dot on top of 3 |
| 3b | 1.428571 need to place bar over 428571 |
| 3c | 1.18 need to place bar over 18 |
| 4a | 3 recurs after the decimal point |
| 4b | The digits 142857 repeat continuously |
| 4c | The digits 18 recur after the decimal point |
| 5a | 5.25 = 5 ¼ = 21/4 |
| 5b | 1/11 |

Chapter 5.1.4 Irrational numbers

|  |  |
| --- | --- |
| 1 | (a), (b), (c), (f) are irrational  (d) is rational as it evaluates to 19  (e) is rational as it can be written as 34/9  (g) is rational |

Chapter 5.1.5 Real numbers

|  |  |
| --- | --- |
| 1a | True |
| 1b | True |
| 1c | True |
| 1d | True |
| 1e | False |
| 1f | True |
| 1g | True |
| 2a | 0.1  1/10 |
| 2b | 0.19 |
| 2c | 0.3439 |
| 2d | A number very close to 1 |
| 2e | 1  0.9999...  (or you could argue 1 – Planck distance) |
| P1 | **Python 3**  **n = int(input("Enter value for n: "))**  **pi = 3**  **for count in range(1, n):**  **x = 4/((count\*2)\*(count\*2+1)\*(count\*2+2))**  **if count % 2 == 0:**  **pi += x**  **else:**  **pi -= x**  **print("The calculation for pi using", n, "terms is", pi)** |
| P1a |  |
| P1b |  |
| P1c |  |
| P1d |  |
| P2 | **Python 3**  **def factorial(n):**  **if n > 1:**  **return n \* factorial(n-1)**  **else:**  **return 1**  **n = int(input(“Enter a value for n: “))**  **e = 0**  **for count in range(0, n + 1):**  **e += (2 \* count + 2) / factorial(2 \* count + 1)**  **print (e)** |
| P2a |  |
| P2b |  |
| P2c |  |
| P2d |  |

Chapter 5.1.6 Ordinal numbers

|  |  |
| --- | --- |
| 1ai | 99 |
| 1aii | 99 |
| 1aiii | 100 |
| 1bi | 99 |
| 1bii | 100 |
| 2 | 0 |
| 3 | 6  The length of the range is one greater than when the < sign is used because both the upper and lower bound of the range are now included i.e. one of the endpoints is now increased/decreased by 1. |
| 4 | Discussion could include the points:   * Some problems are more naturally solved using a different starting point * Having different starting points could make program code harder to understand |
| P1 | **Python 3**  **Letter = input("Enter a letter:").upper()**  **Position = ord(Letter) - 64**  **if Position % 10 == 1:**  **print(str(Position) + "st")**  **elif Position % 10 == 2:**  **print(str(Position) + "nd")**  **elif Position % 10 == 3:**  **print(str(Position) + "rd")**  **else:**  **print(str(Position) + "th")** |

Chapter 5.1.7 Counting and measurement

|  |  |
| --- | --- |
| 1 | 0.1111... |
| 2 | (a) 11/100  (b) 111/1000  (c) 111111/1000000 |
| 3 | (a) 1/900  (b) 1/9000  (c) 1/9000000 |
| P1 | **Python 3**  **print (1 / 3)**  **print (1 / 7)**  **print (1 / 5)**  **print (1 / 10)**    **The first two are not precise and have a (small) error in the representation, the last two are precise representations.** |
| Q4 | (a) 0.3245 and 0.3246  (b)85.994 and 85.995 |
| Q5 | (a) 0.3246  (b) 85.994  (c) 5.884 |

5.2 Number bases

Chapter 5.2.1 Number base

|  |  |
| --- | --- |
| 1a | 1 x 8 + 0 x 4 + 1 x 2 + 0 x 1 |
| 1b | 1 x 8 + 1 x 4 + 1 x 2 + 1 x 1 + 1 x ½ + 1 x ¼ |
| 1c | 1 x 2 + 0 x 1 + 0 x ½ + 1 x ¼ + 0 x 1/8 + 1 x 1/16 |
| 2a | 10 |
| 2b | 15 |
| 2c | 151 |
| 2d | 255 |
| 3a | 2.5 or 2½ |
| 3b | 15/16 |
| 3c | 4 23/32 |
| 3d | 1 127/128 |
| 4a | 1 x 4096 + 0 x 256 + 2 x 16 + 3 x 1 |
| 4b | 1 x 16 + 15 x 1 |
| 4c | 15 x 1 + 1 x 1/16 + 3 x 1/256 |
| 5a | 4131 |
| 5b | 31 |
| 5c | 65535 |
| 5d | 57005 |
| 6a | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **20** | **1** | **1** | **16** | | **2** | **21** | **2** | **0** | **8** | | **3** | **22** | **4** | **0** | **4** | | **4** | **23** | **8** | **0** | **2** | | **5** | **24** | **16** | **0** | **1** | | **6** | **25** | **32** | **1** | **0** |   **So answer is 1000012** |
| 6b | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **20** | **1** | **0** | **12** | | **2** | **21** | **2** | **0** | **6** | | **3** | **22** | **4** | **0** | **3** | | **4** | **23** | **8** | **1** | **1** | | **5** | **23** | **16** | **1** | **0** |   **So answer is 110002** |
| 6c | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **20** | **1** | **0** | **29** | | **2** | **21** | **2** | **1** | **14** | | **3** | **22** | **4** | **0** | **7** | | **4** | **23** | **8** | **1** | **3** | | **5** | **24** | **16** | **1** | **1** | | **6** | **25** | **32** | **1** | **0** |   **So answer is 1110102** |
| 6d | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **20** | **1** | **1** | **63** | | **2** | **21** | **2** | **1** | **31** | | **3** | **22** | **4** | **1** | **15** | | **4** | **23** | **8** | **1** | **7** | | **5** | **24** | **16** | **1** | **3** | | **6** | **25** | **32** | **1** | **1** | | **7** | **26** | **64** | **1** | **0** |   **So answer is 11111112** |
| 7a | |  |  |  | | --- | --- | --- | | **Quotient** | **New number** | **Remainder** | | **33/2** | **16** | **1** | | **16/2** | **8** | **0** | | **8/2** | **4** | **0** | | **4/2** | **2** | **0** | | **2/2** | **1** | **0** | | **1/2** | **0** | **1** |   **So answer is 1000012.** |
| 7b | |  |  |  | | --- | --- | --- | | **Quotient** | **New number** | **Remainder** | | **24/2** | **12** | **0** | | **12/2** | **6** | **0** | | **6/2** | **3** | **0** | | **3/2** | **1** | **1** | | **1/2** | **0** | **1** |   **So answer is 110002.** |
| 7c | |  |  |  | | --- | --- | --- | | **Quotient** | **New number** | **Remainder** | | **58/2** | **29** | **0** | | **29/2** | **14** | **1** | | **14/2** | **7** | **0** | | **7/2** | **3** | **1** | | **3/2** | **1** | **1** | | **1/2** | **0** | **1** |   **So answer is 1110102.** |
| 7d | |  |  |  | | --- | --- | --- | | **Quotient** | **New number** | **Remainder** | | **127/2** | **63** | **1** | | **63/2** | **31** | **1** | | **31/2** | **15** | **1** | | **15/2** | **7** | **1** | | **7/2** | **3** | **1** | | **3/2** | **1** | **1** | | **1/2** | **0** | **1** |   **So answer is 11111112.** |
| 8a | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **160** | **1** | **F (15)** | **2** | | **2** | **161** | **16** | **3** | **0** |   **So answer is 2F16** |
| 8b | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **160** | **1** | **E (14)** | **18** | | **2** | **161** | **16** | **2** | **1** | | **3** | **162** | **256** | **1** | **0** |   **So answer is 12E16** |
| 8c | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **160** | **1** | **D (13)** | **4094** | | **2** | **161** | **16** | **E (14)** | **255** | | **3** | **162** | **256** | **F (15)** | **15** | | **4** | **163** | **4096** | **F (15)** | **0** |   **So answer is FFED16** |
| 8d | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Pass number** | **Multiplier** | | **Digit** | **k at end of pass** | | **1** | **160** | **1** | **A (10)** | **17847** | | **2** | **161** | **16** | **7** | **1115** | | **3** | **162** | **256** | **B (11)** | **69** | | **4** | **163** | **4096** | **5** | **4** | | **5** | **164** | **65536** | **4** | **0** |   **So answer is 45B7A16** |
| 9a | 0100 0111 |
| 9b | 0011 1010 0010 |
| 9c | 0110 1111 1110 0111 |
| 9d | 1011 1110 1110 1111 |
| 10a | F |
| 10b | AD |
| 10c | 2C |
| 10d | CE3 |

5.3 Units of information

Chapter 5.3.1 Bits and bytes

|  |  |
| --- | --- |
| 1 | * A quarter length musical note (crotchet) * An eighth length musical note (quaver) * Happy * Male/man or Helium * Argon * 50% battery charge left |
| 2 | * Road narrows ahead * Crossroad ahead * Roundabout ahead |
| 3 | |  |  |  | | --- | --- | --- | | No of coins | Alphabet | Bits of information  per symbol | | 1 | 2 equiprobable symbols  (h), (t) | 1 | | 2 | 4 equiprobable symbols  (hh), (ht), (th), (tt) | 2 | | 3 | 8 equiprobable symbols  (hhh), (**hht**), (**hth**), (**htt**)  (thh), (**tht**), (tth), (ttt) | **3** | | 4 | 16 equiprobable symbols  (hhhh), (**hhht**), (hhth), (**hhtt**)  (hthh), (**htht**), (**htth**), (**httt**)  (thhh), (**thht**), (**thth**), (**thtt**)  (tthh), (**ttht**), (**ttth**), (tttt) | **4** | |
| T1 | Create a class that has an integer attribute. When attempting to assign a value to the attribute using a method from the class only the values 0-255 will be accepted. |
| 4a | **32**  **25** |
| 4b | **256**  **28** |
| 4c | **65536**  **216** |
| 4d | **16777216**  **224** |
| 4e | **4294967296**  **232** |
| 5a | **28** |
| 5b | **216** |
| 5c | **264** |
| 6 | |  |  | | --- | --- | | **Decimal number** | **Bit pattern** | | 0 | 0000 | | 1 | 0001 | | 2 | 0010 | | 3 | 0011 | | 4 | 0100 | | 5 | 0101 | | 6 | 0110 | | 7 | 0111 | | 8 | 1000 | | 9 | 1001 | | 10 | 1010 | | 11 | 1011 | | 12 | 1100 | | 13 | 1101 | | 14 | 1110 | | 15 | 1111 | |

Chapter 5.3.2 Units

|  |  |
| --- | --- |
| 1a | **23** |
| 1b | **26** |
| 1c | **215** |
| 2a | **210** |
| 2b | **29** |
| 2c | **211** |
| 2d | **212** |
| 2e | **221** |
| 3a | 1,048,576 |
| 3b | 1,536 |
| 3c | 1,879,048,192 |
| 4a | 1 |
| 4b | 0.5 |
| 4c | 2 |
| 4d | 4 |
| 5a | 1 |
| 5b | 6 |
| 5c | 4.5 |
| 5d | 9 |
| 6a | **103** |
| 6b | **106** |
| 6c | **107** |
| 7a | 1kB |
| 7b | 10kB |
| 8a | 0.5MB |
| 8b | 2MB |
| 8c | 30MB |
| 9a | 1,000,000 b/s |
| 9b | 100,000 b/s |
| 9c | 1,000,000,000 b/s |

5.4 Binary number system

Chapter 5.4.1 Unsigned binary

|  |  |
| --- | --- |
| 1a | 00000101 |
| 1b | 10000001 |
| 1c | 11111101 |
| 2a | 161 |
| 2b | 122 |
| 2c | 255 |
| 3a | **Binary: 111111**  **Decimal: 63 or 26-1** |
| 3b | **Binary: 1111111111**  **Decimal: 1023 or 210-1** |
| 3c | **Binary: 1111111111111111**  **Decimal: 65535 or 216-1** |

Chapter 5.4.2 Unsigned binary arithmetic

|  |  |
| --- | --- |
| 1a | 0111 |
| 1b | 1010 |
| 2a | 10000110 |
| 2b | 100110010 |
| 3a | **101**  **10 x**  **----**  **000**  **101**  **----**  **1010** |
| 3b | **101**  **11 x**  **----**  **101**  **101**  **----**  **1111** |
| 3c | **1001**  **10 x**  **-----**  **0000**  **10010**  **-----**  **10010** |
| 3d | **1001**  **11 x**  **-----**  **1001**  **1001**  **-----**  **11011** |
| 3e | **1001**  **101 x**  **------**  **1001**  **0000**  **1001**  **------**  **101101** |
| 3f | **1001011**  **1101 x**  **----------**  **1001011**  **0000000**  **1001011**  **1001011**  **----------**  **1111001111** |
| 3g | **1011100**  **1010 x**  **----------**  **0000000**  **1011100**  **0000000**  **1011100**  **----------**  **1110011000** |
| 3h | **10111101**  **11101 x**  **-------------**  **10111101**  **00000000**  **10111101**  **10111101**  **10111101**  **-------------**  **1010101101001** |

Chapter 5.4.3 Signed binary using two’s complement

|  |  |
| --- | --- |
| 1a | -4 |
| 1b | -16 |
| 1c | -128 |
| 1d | -512 |
| 1e | -32768 |
| 2a | **-22** |
| 2b | **-24** |
| 2c | **-27** |
| 2d | **-28** |
| 2e | **-215** |
| 3a | 100 |
| 3b | 10000 |
| 3c | 10000000 |
| 3d | 1000000000 |
| 3e | 1000000000000000 |
| 4a | 011 |
| 4b | 01111 |
| 4c | 01111111 |
| 4d | 0111111111 |
| 4e | 0111111111111111 |
| 5a | 01100 |
| 5b | 10100 |
| 5c | 00111 |
| 5d | 11001 |
| 5e | 11111 |
| 6a | 00001100 |
| 6b | 11110100 |
| 6c | 11111001 |
| 6d | 00100000 |
| 6e | 11100000 |
| 6f | 10000000 |
| 6g | 11111111 |
| 6h | 11000001 |
| 6i | 10110100 |
| 7a | 92 |
| 7b | -92 |
| 7c | -64 |
| 7d | -1 |
| 7e | -128 |
| 7f | 127 |
| 8a | 0111 - 0100  = 0111 + 1100  = 0011 |
| 8b | 0100 - 1110  = 0100 + 0010  = 0110 |
| 8c | 1101 - 1110  = 1101 + 0010  = 1111 |
| 8d | 1111 - 1100  = 1111 + 0100  = 0011 |
| 8e | 1100 - 0011  = 1100 + 1101  =1001 |
| 9a | **-23 to 23-1**  **-8 to 7** |
| 9b | **-25 to 25-1**  **-32 to 31** |
| 9c | **-29 to 29-1**  **-512 to 511** |
| 9d | **-215 to 215-1**  **-32768 to 32767** |

Chapter 5.4.4 Numbers with a fractional part

|  |  |
| --- | --- |
| 1a | 3½ |
| 1b | 5¾ |
| 1c | 16 5/8 |
| 1d | 31 7/8 |
| 2a | 1 1/16 |
| 2b | 2 3/16 |
| 2c | 15¼ |
| 2d | 10 7/16 |
| 3a | 32 1/64 |
| 3b | 56 1/32 |
| 3c | 15 3/64 |
| 3d | 49 7/64 |
| 4a | -3½ |
| 4b | -5¾ |
| 4c | -8 5/8 |
| 4d | -3 7/8 |
| 5a | Mantissa: 0100  Exponent: 0100  Decimal expansion: 0.5 x 24 = 8 |
| 5b | Mantissa: 1010  Exponent: 0100  Decimal expansion: -0.75 x 24 = -12 |
| 5c | Mantissa: 0100  Exponent: 1111  Decimal expansion: 0.5 x 2-1 = 0.25 |
| 5d | Mantissa: 0111  Exponent: 0011  Decimal expansion: 7/8 x 23 = 7 |
| 5e | Mantissa: 1000  Exponent: 0000  Decimal expansion: -1 x 20 = -1 |
| 5f | Mantissa: 1111  Exponent: 1100  Decimal expansion: -1/8 x 2-4 = -1/128 |
| 6a | |  |  |  | | --- | --- | --- | | **F** | **R** | **D** | | 0.375 | 0.75 | 0 | | 0.75 | 1.5 | 1 | | 0.5 | 1.0 | 1 | | 0 |  |  |   0.011 |
| 6b | |  |  |  | | --- | --- | --- | | **F** | **R** | **D** | | 0.4 | 0.8 | 0 | | 0.8 | 1.6 | 1 | | 0.6 | 1.2 | 1 | | 0.2 | 0.4 | 0 | | 0.4 |  |  | |
| 6c | |  |  |  | | --- | --- | --- | | **F** | **R** | **D** | | 0.7 | 1.4 | 1 | | 0.4 | 0.8 | 0 | | 0.8 | 1.6 | 1 | | 0.6 | 1.2 | 1 | | 0.2 | 0.4 | 0 | | 0.4 |  |  | |
| 6d | |  |  |  | | --- | --- | --- | | **F** | **R** | **D** | | 0.703125 | 1.40625 | 1 | | 0.40625 | 0.8125 | 0 | | 0.8125 | 1.625 | 1 | | 0.625 | 1.25 | 1 | | 0.25 | 0.5 | 0 | | 0.5 | 1.0 | 1 | | 0 |  |  | |
| 6e | |  |  |  | | --- | --- | --- | | **F** | **R** | **D** | | 0.1 | 0.2 | 0 | | 0.2 | 0.4 | 0 | | 0.4 | 0.8 | 0 | | 0.8 | 1.6 | 1 | | 0.6 | 1.2 | 1 | | 0.2 |  |  | |
| 7a | |  |  |  | | --- | --- | --- | | **F** | **R** | **D** | | 0.875 | 1.75 | 1 | | 0.75 | 1.5 | 1 | | 0.5 | 1.0 | 1 | | 0 |  |  |  |  |  | | --- | --- | | 50 | 1 | | 25 | 0 | | 12 | 1 | | 6 | 0 | | 3 | 0 | | 1 | 1 | | 0 | 1 |   1100101.111 |
| 7b | |  |  |  | | --- | --- | --- | | **F** | **R** | **D** | | 0.55 | 1.1 | 1 | | 0.1 | 0.2 | 0 | | 0.2 | 0.4 | 0 | | 0.4 | 0.8 | 0 | | 0.8 | 1.6 | 1 | | 0.6 | 1.2 | 1 | | 0.2 |  |  |  |  |  | | --- | --- | | 166 | 1 | | 83 | 0 | | 41 | 1 | | 20 | 1 | | 10 | 0 | | 5 | 0 | | 2 | 1 | | 1 | 0 | | 0 | 1 | |
| 8a | 11110.110 STOPPED HERE |
| 8b | 11000.100 |
| 8c | 11111.000 |
| 8d | 10000.001 |
| 9a | 01010 001 |
| 9b | 10101 011 |
| 9c | 01111 011 |
| 9d | 01000 001 |
| 9e | 10000 000 |
| 9f | Can’t be represented exactly in the allowed number of bits, 01110 011 is as close as you can get |
| 10a | 80 |
| 10b | -64 |
| 10c | 1/512 |
| 10d | 7/256 |
| 10e | -7/16 |
| T1 | **Answer written using Pascal (Delphi XE5 console mode)**  **Program Ch4Pt4T1;**  **{$APPTYPE CONSOLE}**  **{$R \*.res}**  **Uses**  **System.SysUtils;**  **Var**  **n, W, D, AllocatedFractionalNoOfBits : Integer;**  **OrigF, R, DecimalNo, F : Real;**  **Begin**  **Try**  **Write('Input decimal number to be converted: ');**  **Readln(DecimalNo);**  **Write('How many bits? ');**  **Readln(AllocatedFractionalNoOfBits);**  **W := Trunc(DecimalNo);**  **F := DecimalNo - W;**  **n:= 0;**  **OrigF := F;**  **Repeat**  **R:= F \* 2;**  **D := Trunc(R);**  **Write(D);**  **n:= n + 1;**  **F := R - D;**  **Until (F = 0) Or (F = OrigF)**  **Or (n = AllocatedFractionalNoOfBits);**  **Except**  **On E: Exception**  **Do Writeln(E.ClassName, ': ', E.Message);**  **End;**  **Readln;**  **End.**  **Answer written using Python 3**  OrigF = float(input(**"Enter fractional part: "**)) AllocatedFractionalNoOfBits = int(input(**"Enter allocated number of fractional bits: "**)) n = 0 F = 1.0 R = OrigF \* 2.0 D = 0 print (**"0."**, end=**""**) **while** F !=0 **and** F != OrigF **and** n != AllocatedFractionalNoOfBits:  **if** R >= 1:  D = 1  **else**:  D = 0  print(D, end=**""**)  n = n + 1  F = R - D  R = F \* 2.0 |

Chapter 5.4.5 Rounding errors

|  |  |
| --- | --- |
| 1a | (3 x 100) + (0 x 10) + (2 x 1) + (0 x 1/10) + (3x 1/100) + (4 x 1/1000) |
| 1b | (5 x 1000) + (1 x 100) + (2 x 10) + (0 x 1) + (2 x 1/10) + (0 x 1/100) + (0 x 1/1000) + (7 x 1/10000) |
| 1c | (0 x 1) + (4 x 1/10) + (5 x 1/100) + (6 x 1/1000) + (7 x 1/10000) |
| 2a | (1 x 8) + (1 x 4) + (0 x 2) + (0 x 1) + (1 x 1/2) + (1 x 1/4) |
| 2b | (1 x 4) + (0 x 2) + (1 x 1) + (0 x ½) + (1 x ¼) + (0 x 1/8) + (1 x 1/16) |
| 2c | (1 x 2) + (1 x 1) + (1 x ½) + (0 x ¼) + (1 x 1/8) + ( 1 x 1/16) |
| 3a | **51 / 22** |
| 3b | **85 / 24** |
| 3c | **59 / 24** |
| 4a | 1 / (2 x 5) |
| 4b | 7 / (2 x 5) |
| 4c | (3 x 19) / (2 x 10) |
| 4d | (5 x 7) / (2 x 2) |
| 4e | (3 x 5 x 11 x 13) / (2 x 2 x 2 x 2 x 2) |
| 5a | No |
| 5b | No |
| 5c | No |
| 5d | Yes |
| 5e | Yes |
| 6a | **there are prime factors in the denominator other than 2 so it can’t be represented exactly in fixed point binary** |
| 6b | **there are prime factors in the denominator other than 2 so it can’t be represented exactly in fixed point binary** |
| 6c | **there are prime factors in the denominator other than 2 so it can’t be represented exactly in fixed point binary** |
| 6d | **1000.11 the only prime factor in the denominator is 2 so it can be represented exactly in fixed point** |
| 6e | **1000011.00001 the only prime factor in the denominator is 2 so it can be represented exactly in fixed point** |
| 7a | 0.0000000 |
| 7b | 0.0000001 |
| 7c | 0.0000010 |
| 7d | 0.0000011 |
| 7e | 0.0000101 |
| 7f | 1.1111110 |
| 8a | 1.1111111 |
| 8b | 1.1111110 |
| 8c | 1.1111111 (not an exact representation) |
| 9a | (1 x 256/256) + (1 x 128/256) + (1 x 64/256) + (1 x 32/256) + (1 x 16/256) + (1 x 8/256) + (1 x 4/256) + (1 x 2/256)  1.9921875  Exact representation |
| 9b | (1 x 256/256) + (1 x 128/256) + (1 x 64/256) + (1 x 32/256) + (1 x 16/256) + (1 x 8/256) + (1 x 4/256) + (0 x 2/256)  1.984375  Exact representation |
| 9c | (1 x 256/256) + (1 x 128/256) + (1 x 64/256) + (1 x 32/256) + (1 x 16/256) + (1 x 8/256) + (1 x 4/256) + (1 x 2/256)  1.9921875  Not an exact representation |
| 10 | |  |  |  |  | | --- | --- | --- | --- | | Fixed point  decimal | Fixed point binary | 10-bit fixed point binary (rounded off) | Rounding  error | | 0.1 | 0.0 | 0.000110011 | 0.000390625 | | 0.2 | 0. | 0.001100110 | 0.00078125 | | 0.25 | 0.01 | 0.010000000 | 0 | | 0.3 | 0.0 | 0.010011001 | 0.00117175 | | 0.4 | 0. | 0.011001101 | 0.000390625 | | 0.5 | 0.1 | 0.100000000 | 0 | | 0.6 | 0. | 0.100110011 | 0.00030625 | | 0.7 | 0.1 | 0.101100110 | 0.00078125 | | 0.75 | 0.11 | 0.110000000 | 0 | | 0.8 | 0. | 0.110011001 | 0.001171875 | | 0.9 | 0.1 | 0.111001101 | 0.000390625 | | 1.0 | 1.0 | 1.000000000 | 0 | |

Chapter 5.4.6 Absolute and relative errors

|  |  |
| --- | --- |
| 1a | The representation would be 0.0011010 (0.0011001100110011...... rounded off)  Absolute error is 0.2 – 0.203125 = 0.003125 |
| 1b | The representation would be 0.1001101 (0.10011001100110011...... rounded off)  Absolute error would be 0.6 – 0.6015625 = 0.0015625 |
| 2a | Absolute error is 0.2 – 0.203125 = 0.003125  Relative error is 0.003125/0.2 = 0.015625 which is 1.5625% |
| 2b | Absolute error would be 0.6 – 0.6015625 = 0.0015625  Relative error is 0.0015625/0.6 = 0.0026041666... which is 0.26042% to 5 sig figs |
| 3a | **0.01 or 1.00 x 10-2** |
| 3b | **1.00 x 1036** |
| 3c | **1.00 x 10-41** |

Chapter 5.4.7 Range and precision

|  |  |
| --- | --- |
| **1a** | **1/8** |
| **1b** | **1/32768** |
| **1c** | **1/8388608** |
| **1d** | **1/2147483648** |
| **1e** | **1/9223372036854775808** |
| **2a** | **14** |
| **2b** | **40** |
| **2c** | **-2.5** |
| **2d** | **0.009765625**  **or**  **5/512** |
| **3** | **Because each time you increment the exponent you are doubling the range of numbers represented but still have only the same precision in the mantissa.** |
| **4a** | **Most positive:**  **1.75**  **Most negative:**  **-2** |
| **4b** | **Most positive:**  **7 15/16**  **Most negative:**  **-8** |
| **4c** | **Most positive:**  **15 31**/32  **Most negative:**  **-16** |
| **5a** | **Most positive:**  **(1 – 2-7) x 2127**  **or approximately**  **1.7 x 1038**  **Most negative:**  **-1 x 2127**  **or approximately -1.7 x 1038** |
| **5b** | **Most positive:**  **(1 – 2-15) x 232767**  **or approximately**  **7.1 x 109863**  **Most negative:**  **-1 x 232767**  **or approximately**  **-7.1 x 109863** |
| **5c** | **Most positive:**  **(1 – 2-31) x 22147483647**  **Most negative:**  **-1 x 22147483647** |
| **6a** | **4** |
| **6b** | **5** |
| **6c** | **1** |
| **6d** | **3** |
| **6e** | **Ambiguous** |
| **6f** | **3** |
| **7a** | **3** |
| **7b** | **3** |
| **7c** | **2** |
| **7d** | **3** |
| **7e** | **7** |
| **7f** | **Ambiguous** |
| **8a** | **9** |
| **8b** | **25** |
| **8c** | **55** |
| **9a** | **16** |
| **9b** | **24** |
| **9c** | **64** |

Chapter 5.4.8 Normalisation of floating point form

|  |  |
| --- | --- |
| **1a** | **0.0000323142 x 107** |
| **1b** | **0.000000323142 x 109** |
| **1c** | **32314200 x 10-5** |
| **2a** | **+4.56789 +2** |
| **2b** | **+4.56789 -1** |
| **2c** | **+4.56789 -8** |
| **2d** | **+4.56789 +1** |
| **3a** | **0.11101 1111** |
| **3b** | **0.10101 1010** |
| **3c** | **0.11101 0001** |
| **4a** | **1.01** |
| **4b** | **101.011** |
| **4c** | **011.01** |
| **4d** | **101.01** |
| **4e** | **101.0100** |
| **5a** | **1.01110 1111** |
| **5b** | **1.01011 1010** |
| **5c** | **1.00010 0001** |

Chapter 5.4.9 Underflow and overflow

|  |  |
| --- | --- |
| **1a** | **Yes** |
| **1b** | **No** |
| **1c** | **No** |
| **1d** | **Yes** |
| **2** | **Multiplication first then the division as this means that underflow is less likely to occur** |
| **3a** | **Yes** |
| **3b** | **Yes** |
| **3c** | **Yes** |
| **3d** | **No** |
| **4a** | **Yes** |
| **4b** | **No** |
| **4c** | **Yes** |
| **4d** | **Yes** |
| **5** | **Division first then multiplication so overflow is less likely to occur.** |
| **6** | **If one of the operands is large then it is better to use the right hand side in order to avoid overflow that could result from squaring.** |

5.5 Information coding systems

Chapter 5.5 Information coding systems

|  |  |
| --- | --- |
| 1a) | 72 |
| 1b) | 51 |
| 1c) | 63 |
| 2a) | a |
| 2b) | % |
| 2c) | 0 |
| 3) | 72 101 108 108 111 |
| 4) | Undoing holes in paper tape was not feasible so if a character’s set of holes needed to be ignored, for example, because they were no longer valid then the paper tape reader needed to know this. The easiest solution was to reserve a particular pattern of holes for this purpose. The line of 7 holes across the tape was chosen because every other character punched as holes on the tape was easily turned into the 7 holes pattern using a pin of suitable diameter to punch the extra holes where required. The 7 holes pattern corresponds to 11111112 or 12710. |
| 5) | 72 101 108 108 111 13 10 87 111 114 108 100 33 |
| 6) | Hello  World! |
| 7a) | Subtract 48 or AND with 01100002 |
| 7b) | Subtract 48 or AND with 01100002 |
| 7c) | Subtract 48 or AND with 01100002 |
| 8a) | 54 |
| 8b) | 51 52 |
| 8c) | 57 48 56 |
| 8d) | 52 52 52 |
| 9) | Because they do not contain all the information about the magnitude of the various digits i.e. there is no information about if the digit 3 represents 3 units, 3 tens, etc…  Also, just adding the codes does not result in the code for the sum of the two numbers e.g. 49 + 50 (1+2) is 99 and does not give the ASCII code for 3 (51). |
| 10) | Subtract 48 from each character code and then multiply each digit by a power of 10 and sum the results e.g. 49 48 50 🡪 1 0 2 🡪 1 \* 103 + 0 \* 102 + 2 \* 101 🡪 102 |
| 11a) | 54 |
| 11b) | 53 52 |
| 12a) | 1 |
| 12b) | 0 |
| 13a) | 0 |
| 13b) | 1 |
| 14) | Using XOR the parity bit is calculated from the data bits received. This parity bit is XORed with the parity bit received. If the result of this is a 1 then there has been an error affecting an odd number of bits.  Or  The number of 1s in the received data bits is counted and the parity bit regenerated. The regenerated parity bit is compared with the received parity bit. If they disagree then an error has occurred affecting an odd number of bits. |
| 15) | A checksum is a number calculated from a block of data and then appended to the block of data for error detection and correction purposes. |
| 16) | The bit in the 4th row, 4th column contains the error. |
| 17) | A check digit is a digit calculated from the digits of a number and then added to the number for validation purposes. |
| 18) | The three most common mistakes are:   * transposing two digits * missing out / adding in a single digit * changing a single digit   A mistake in a single digit can be identified using modulo arithmetic. One way of doing this is to sum the digits of the number and then add on a value (c) so that when the mod operator is applied to the sum using a chosen value (p) the result is 0. The value c can be used as a check digit.  To adapt this method so that it will also detect the transposition of two digits each digit should be multiplied by a different weight prior to summing them up. |

5.6 Representing images, sound and other data

Chapter 5.6.1(1), (2), (3) Bit patterns, images, sound and other data

|  |  |
| --- | --- |
| 1) | Because computer hardware uses the language of binary. |
| 2) | A group of bits. |
| 3) | Divide the scene into a two-dimensional grid. Measure the brightness of the colour in each cell in the grid. Represent each of these values as a discrete value. |
| 4) | Metadata will also have been stored about the image. Metadata is needed as it includes information about the image, e.g. bits per pixel, needed for successful processing of the image data. |
| 5) | Multiply both values by 0.5.  C(:,1,1) = 0.5 \* C(:,1,1); |
| 6) | Change the sampling rate e.g. if the sampling rate was 20000 samples per second then changing this value to 40000 would result in the frequency doubling. |
| 7) | Export data to a spreadsheet or to other applications  Data can be easily manipulated by the user  Text files can be used on multiple platforms |

Chapter 5.6.2 Analogue and digital

|  |  |
| --- | --- |
| 1) | Data that varies in a continuous manner or is recorded in a continuous form. |
| 2) | Data which is in discrete form. Examples: the number of people waiting at a bus stop, the number of books on a book shelf. |
| 3) | A signal is a sound/gesture/action/etc... that conveys a message or information from one place to another. |
| 4) | An analogue signal is a signal that varies in a continuous manner and a digital signal is a signal in which the changes of value happen in discrete steps. |

Chapter 5.6.3 Analogue/digital conversion

|  |  |
| --- | --- |
| 1) | A single measure of amplitude. |
| 2a) | 20,000 |
| 2b) | 40,000 |
| 2c) | 44,100 |
| 3) | The analogue signal is sampled at regular intervals. The amplitude of each sample is measured and represented using a fixed number of bits. The binary form of the sample is represented using a series of electrical pulses that can be transmitted via a bus. |
| 4) | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | -9 | -8 | | -7 | | -6 | | -5 | | -4 | | -3 | | -2 | | -1 | | 0 | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | | 100 | | | | 101 | | | | 110 | | | | 111 | | | | 000 | | | | 001 | | | | 010 | | | | 011 | | | | |
| 5a) | 001 |
| 5b) | 001 |
| 5c) | 111 |
| 5d) | 011 |
| 6) | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0.5 | | 1 | | 1.5 | | 2 | | 2.5 | | 3 | | 3.5 | | 4 | | 4.5 | | 5 | | 5.5 | | 6 | | 6.5 | | 7 | | 7.5 | | 8 | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | | | 000 | | | | 001 | | | | 010 | | | | 011 | | | | 100 | | | | 101 | | | | 110 | | | | 111 | | | | |
| 7a) | 1/8 3/8 5/8 7/8 11/8 1 3/8 1 5/8 1 7/8 2 1/8 2 3/8 2 5/8 2 7/8 3 1/8 3 3/8 3 5/8 3 7/8  0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 |
| 7b) | 4 |
| 7c) | 0.25 |
| 7d) | 0.125 (1/8) |
| 8) | 10 \* 40,000 \* 180 / 8 = 9,000,000 bytes |
| 9) | 3 \* 60 = 180 seconds  180 \* 44,100 \* 16 / 8 = 15,876,000 bytes per channel per 3 minute recording  15,876,000 \* 2 = 31,752,000 bytes for a two-channel stereo 3 minute recording  737,000,000 / 31,752,000, so 23 recordings can be stored on the CD-ROM. |
| 10) | To convert a digital signal to an analogue signal (that approximates the original analogue signal from which the digital signal was derived). |
| 11) | To convert digital audio to analogue. |
| 12) | A sensor is a device that measures something of interest in the physical world; it usually has a transducer to convert what is sensed  into an equivalent electrical signal.  An ADC is needed as the output from a sensor is an analogue signal which needs to be converted into a digital signal so that it can  be processed by a computer. |
| 13) | Examples include: gyroscope, accelerometers, pressure sensors. |
| 14) | ADC  Microcontroller  I/O bus  Sensor |
| 15) | Digital format is not suitable to playback through loudspeakers as it will not sound like the original analogue sound from which  it was derived. DAC converts the digital audio into an approximate analogue equivalent which can be played back through loudspeakers. |
| 16) | The PCM signal is converted into a PAM signal.  A smoothing algorithm is applied to the PAM signal.  See figures 5.6.3.8 and 5.6.3.9 for diagrams that could be used to aid the explanation. |
| 17) | Fewer bits were used to represent each sample taken from the original analogue signal. Even though the sampling rate used  may have been the same the lower sampling resolution means that the maximum quantisation error that could occur for each  sample is considerably higher when the 8-bit system was used so each measurement is potentially not as accurate as the  equivalent measurement taken using the 16-bit system. This means that when the sound is played back it may be of noticeably  lower-quality that the sound from the 16-bit system. |

Chapter 5.6.4 Bitmapped graphics

|  |  |
| --- | --- |
| 1 | The smallest addressable element of a digital image. |
| 2 | Sampling is digitising coordinate values in an image – it takes a continuous image and creates an array of discrete values that approximate the digital image.  Quantisation is the process of taking the sampled values and representing each of them as one of a finite set of values (bit patterns). |
| 3(a) | 720 x 480 = 345,600 |
| 3(b) | 1920 x 1280 = 2,457,600 |
| 3(c) | 5184 x 3456 = 17,915,904 |
| 4 | 17.9 (to 1d.p.) |
| 5(a) | The table below shows how the image could be represented as a bitmap:   |  |  | | --- | --- | | Cell | Contents | | 307 | 01111101 | | 308 | 11111111 | | 309 | 00000000 | | 310 | 11111111 | | 311 | 01111101 | | 312 | 11111111 | | 313 | 01111101 | | 314 | 11111111 | | 315 | 11111111 | | 316 | 00000000 | | 317 | 11111111 | | 318 | 01111101 | | 319 | 11111111 | | 320 | 00000000 | | 321 | 11111111 | | 322 | 01111101 | |
| 5(b) | 224 |
| 6(a) | 212 |
| 6(b) | 215 |
| 6(c) | 218 |
| 7(a) | 2560/11.3 or 1600/7.04 = 227ppi |
| 7(b) | 2160/10 or 1440/6.67 = 216ppi |
| 7(c) | 1920/20.43 or 1200/12.77 = 94ppi |
| 7(d) | 2560/5.19 or 1440/2.92 = 493ppi |
| 8 | The answers to question 7 show that larger screens (like the one in 7c) can have a much lower pixel density than small screens (like the one in 7d). Small screens e.g. smartphone tend to be viewed at a much shorter distance than large screens (e.g. for desktop computers) and so need a higher pixel density for the display quality to be considered acceptable. |
| 9 | Small screens e.g. smartphone tend to be viewed at a much shorter distance than large screens (e.g. for desktop computers) and so need a higher pixel density for the display quality to be considered acceptable. |
| 10 | Because at levels below this the human eye can discern the loss of quality when the photos are printed at a standard size. |
| 11 | 2.13 x 1.60 inches |
| 12 | 1260 x 960 pixels |
| 13(a) | 300 x 16 = 4800 pixels |
| 13(b) | 0.94 x n samples |
| 13(c) | 0.94 x n = 300 x 16, therefore n = (300 x 16) / 0.94 = 5107 samples per inch |
| 13(d) | 4800 x 7252 |
| 14 | File type, width in pixels, height in pixels, bits per pixel, bitmap data offset, type of compression, size in bytes of bitmap, resolution in pixels per metre, number of colours used |
| 15 | 460800 bytes |

Chapter 5.6.5 Vector graphics

|  |  |
| --- | --- |
| 1) | As a set of commands that describe objects and their associated properties. |
| 2) | Because the graphic consists of identifiable objects which are represented by mathematical descriptions in the form: type of object and a list of numbers; this makes each object scalable to fit the resolution of the display device by simply applying an appropriate scaling factor to the list of numbers so each graphic object can be recreated at any resolution. |
| 3) | Because to scale a vector graphic you just apply mathematical operators to the parameter values, with a bitmapped image though each pixel is enlarged making the image distort. |
| 4) | Circle, rectangle, arc |
| 5) | Stroke/line width  Stroke/line colour  Fill colour  Type of object |

Chapter 5.6.6 Vector graphics versus bitmapped graphics

|  |  |
| --- | --- |
| 1) | Objects in a bitmapped image are difficult to move because objects are not stored individually – instead information is stored about each individual pixel. To move an object requires changing the colour of each pixel where the “object” currently is and the colour of each pixel where the “object” is being moved to. It is not normally easy to identify which pixels belong to an “object” and which do not. |
| 2) | Manipulating objects in a vector graphic image is straightforward as all you have to do is to change a parameter value for one of the commands that create the object. |
| 3) | Vector objects are scalable.  Vector objects are reusable.  Vector graphic files are (normally) smaller in size.  Objects in vector graphic images are easier to edit. |
| 4) | There are many images that cannot be accurately constructed using the limited range of object types that vector graphics use.  Bitmapped images are also better at representing textures in a realistic way. |
| 5) | Vector graphics are the best choice when the image can be constructed from shape and text objects e.g. charts, diagrams, cartoons – i.e. drawing tasks. When an image can be represented using shapes and text it can be represented as a vector graphic which makes the image scalable and gives all the other advantages of vector graphics. |
| 6) | Bitmapped graphics are the best choice for capturing an image from the real world (photos) because these types of images cannot be represented using shapes and text. Painting tasks are also better as bitmapped images because they can represent textures in a realistic way. |

Chapter 5.6.7 Digital representation of sound

|  |  |
| --- | --- |
| 1a) | Sampling rate is the number of samples taken per second. |
| 1b) | Sampling resolution is the number of bits used to represent a sample. |
| 2) | The sampling rate chosen is too low to capture all the harmonic frequencies. To get round this problem a sampling rate of at least 20 kHz should be used. |
| 3) | Because the human ear can hear frequencies up to 20,000 Hz and so the sampling rate should used should be at least twice that to ensure that the harmonic frequencies are captured in the digital representation of the sound. If this was not done then harmonics not in the original sound would be included and result in a distortion of the original sound. |
| 4) | Sampling should be done at a rate that is at least twice that of the highest frequency in the waveform being sampled if all the frequencies in the waveform are to be preserved. |

Chapter 5.6.8 Musical Instrument Digital Interface (MIDI)

|  |  |
| --- | --- |
| 1) | It is a hardware and software specification for the exchange of information (musical notes, expression control, etc) between different musical instruments or other devices such as sequencers, computers, lighting controllers, etc.  It is a hardware / software protocol to enable electronic instruments and other digital musical tools to communicate with each other using the same set of agreed- upon codes and numbers. |
| 2) | MIDI is a series of messages – MIDI instruments can be used to interpret these messages and produce a sound. |
| 3) | 144 60 64  The 1st byte is the status byte with the first 4 bits giving the command (Note-On in this example) and the second 4 bits giving the channel (Channel 1 in this example).  The 2nd byte is the note number which is used by the receiving instrument to select the frequency of note to play.  The 3rd byte is the velocity with which the key was pressed. It will be mapped by the receiving instrument to control the loudness of volume of the note played. |
| 4) | Change instrument  Pitch bend  Note Off |
| 5) | A MIDI message is the means by which an event in one system, e.g. key pressed on a keyboard, is communicated or transported to another to produce an event in the receiving system, e.g. a synthesiser plays a note.  Keyboard-event → MIDI message → synthesiser event.  For example, the “Note On” message sent by a MIDI controller to a MIDI instrument causes an event to take place, i.e. the synthesiser plays the note specified in the message by note number.  The fundamental structure of event-handlers is that they have a non-terminating loop that “sleeps” until an event occurs when they then spring into action to deal with the event. |
| 6) | Can easily change the instrument.  Can easily edit the music (e.g. move all notes up an octave).  Requires less storage space.  Can compose and notate algorithmically which is not possible to with sampled audio. MIDI data can easily be generated by code which cannot be done with sampled audio. |

Chapter 5.6.9 Data compression

|  |  |
| --- | --- |
| 1) | Compressing data is reducing the number of bytes that the data would take up if it was uncompressed. |
| 2) | Structured data can be compressed without loss because it contains redundant data caused by the fact that it contains regular patterns. |
| 3) | To reduce the amount of storage space that the data requires and to reduce the time taken to transmit the data. |
| 4) | No. |
| 5) | Different types of data / collections of data have different properties and so require different compression methods to be compressed effectively. |
| 6) | Lossless compression compresses the data in a way that means there is no loss of data and the original (uncompressed) data can be recovered from the compressed data. |
| 7) | Lossy compression removes some irrelevant data during the compression process meaning that the original (uncompressed) data cannot be recovered from the compressed data. |
| 8) | An advantage of lossy compression is that it reduces the number of bytes that the data would take up further than lossless compression. |
| 9) | An advantage of lossless compression is that the original (uncompressed) data can be recovered from the compressed data. |
| 10) | Lossless compression is often used with text data as it is important that the original (uncompressed) data can be recovered. Lossy compression is often used with audio data as there are often harmonics that the human ear is not sensitive enough to notice and these harmonics can be removed from the data as they are considered irrelevant. |
| 11) | Runs of contiguous values are replaced by the count of the number of contiguous values and a single instance of the value. |
| 12) | Before: 15, 112, 112, 112, 98, 76, 76, 15, 46, 46, 46, 46, 46, 19, 101, 6  After: 15, 255 3, 112, 98, 255, 2, 76, 15, 255, 5, 46, 19, 101, 6 |
| 13) | RLE works well for images which have large areas of the same colour as these contain long runs of contiguous values that can be effectively compressed using this method.  RLE does not work well with pictures that contain many colours with few runs of the same colour. |
| 14) | RLE is not normally effective with text as there are rarely long runs of the same character in text data. |
| 15) | In dictionary-based compression systems, common combinations of data values in the data are replaced with shorter codes. A list of data values and their corresponding codes are kept in a dictionary. |
| 16) | |  |  |  | | --- | --- | --- | | 0 | Empty string | Token | | 1 | B | <0, B> | | 2 | A | <0, A> | | 3 | BB | <1, B> | | 4 | AB | <2, B> | | 5 | ABA | <4, A> | | 6 | ABB | <4, B> |   (a)  (b) B A BB AB B BB ABA ABB |
| 17) | <0, B> <1, B> <0, A> <1, B> <3, A> <1, D> <1, A> <2, B>  Firstly, the input will be in the same sequence as the encoded output. The root of the trie/tree is labelled 0. All strings that begin with the empty string are children of the root, so will appear in the encoded output as <0, something>. Therefore, B and A are children of the root.  The first token in the input is <0, B> so we add B to the dictionary at position 1, the first empty slot after Empty string and we label the leftmost leaf node 1 (dictionary index) and label the branch to it B as shown below.    The next token is <1, B> which indicates a block occurred at node labelled 1. The input was BB. We add this string to the dictionary at the next available entry which is 2. We create a node labelled 2 and a branch labelled B which we connect to node 1 as shown below.    The next token is <0, A>. A doesn’t yet exist in the dictionary so we add A at position 3. We add a branch labelled A to the tree, connect it to the root node labelled 0 and add a new node labelled 3 as shown below.    The next token is <1, B>, dictionary entry 1 exists, it is B, so we concatenate B with the rest of the token, i.e. to get the decoded output BB.  We have worked out that the input that produced the encoded tokens processed so far must have been  B BB A BB  The next token <3, A> refers to dictionary entry 3 which is A. So the decoded output is AA. We don’t have a dictionary entry for AA so it is added at the next free position in the dictionary, slot 4. We create a branch labelled A and connect it to node 3 and a new node labelled 4, the dictionary entry index. The new trie is shown below.    The input that produced the encoded tokens processed so far must have been  B BB A BB AA  The remaining tokens <1, D> <1, A> <2, B> decoded are BD BA BBB. Therefore the input that was coded was  B BB A BB AA BD BA BBB  And the final dictionary tree is |

Chapter 5.6.10

|  |  |
| --- | --- |
| 1) | AOL ZBU OHZ NVA PAZ OHA VU |
| 2) | HIP HIP HOORAY |
|  |  |
| 3) | STUDENTS SHOULD BE FAMILIAR WITH THE TERMS CIPHER PLAINTEXT AND CIPHERTEXT CAESAR AND VERNAM CIPHERS ARE AT OPPOSITE EXT REMES ONE OFFERS PERFECT SECRECY THE OTHER DOESNT |
| 4) | a) 0  b) 10  c) 19  d) 10 |
| 5) | a) 20  b) 17  c) 13  d) 18 |
| 6) | a) 12  b) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 |
| 7) | a) Thursday  b) Tuesday |
| 8) | 10 |
| 9) | 320 |
| 10) | a) the whole numbers between 0 and 4 (inclusive) i.e. 0, 1, 2, 3, 4  b) the whole numbers between 0 and 8 (inclusive) i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8 |
| 11) | a) q = 3, r = 1  b) q = 1, r = 14  c) q = 1, r = 16  d) q = 1, r = 211 |
| 12) | a) q = -3, r = 11  b) q = -1, r = 10  c) q = -1, r = 44  d) q = -1, r = 154 |
|  | Progamming Activity 1 (possible answer using Python 3)  l = [["A",0],["B",1],  ["C",2],["D",3],  ["E",4],["F",5],  ["G",6],["H",7],  ["I",8],["J",9],  ["K",10],["L",11],  ["M",12],["N",13],  ["O",14],["P",15],  ["Q",16],["R",17],  ["S",18],["T",19],  ["U",20],["V",21],  ["W",22],["X",23],  ["Y",24],["Z",25],  [" ",26]]  plaintext = "THIS IS A MESSAGE"  key = -1  while key not in range(0,27):  key = int(input("Enter a key (between 0 and 26):"))  print("The ciphertext produced using your key is:", end="")  ciphertext = ""  for c in plaintext:  temp = -1  for sl in l:  if sl[0] == c:  temp = (sl[1] + key) % 27  for sl in l :  if sl[1] == temp:  ciphertext += sl[0]  temp = -1  print (ciphertext) |
|  | Progamming Activity 2 (possible answer using Python 3)  l = [["A",0],["B",1],  ["C",2],["D",3],  ["E",4],["F",5],  ["G",6],["H",7],  ["I",8],["J",9],  ["K",10],["L",11],  ["M",12],["N",13],  ["O",14],["P",15],  ["Q",16],["R",17],  ["S",18],["T",19],  ["U",20],["V",21],  ["W",22],["X",23],  ["Y",24],["Z",25],  [" ",26]]  ciphertext = "UIJTAJTABANFTTBHF"  key = -1  while key not in range(0,27):  key = int(input("Enter a key (between 0 and 26):"))  print("The plaintext produced using your key is:", end="")  plaintext = ""  for c in ciphertext:  temp = -1  for sl in l:  if sl[0] == c:  temp = (sl[1] - key) % 27  for sl in l :  if sl[1] == temp:  plaintext += sl[0]  temp = -1  print (plaintext) |
| 13) | Because there are only 27 different characters used in the original plaintext. |
| 14) | Because there are only 27 different characters used in the original plaintext. |
| 15) | She could try each of the possible negative values between -1 and -26 on the first letter of the ciphertext until she finds the one that produces the “D” of the phrase “DEAR BOB” at the start. The absolute value of the negative value that produced the phrase is the key that was used. |
| 16) | Because even though Eve is fairly sure that the first few characters represent the phrase “DEAR BOB” this does not help her to work out any of the rest of the text as the same plaintext character is not always represented by the same ciphertext character. |
| 17) | 110001000101110 |
| 18) | Using c1 ⊕ c2 = m1 ⊕ m2  Exclusive-Or c1 and c2 to get m1 ⊕ m2  010100001010110011001000111110111101000011101110  010000011010010011011101100100111101110011110011  ---------------------------------------------------------------------------  000100010000100000010101011010000000110000011101  Separate this result into 8-bit chunks  00010001#00001000#00010101#01101000#00001100#00011101  Extract the bytes that begin 01  01101000  Therefore, for this byte m1 ⊕ m2 = 01101000  Because the fourth byte begins with 01 we know that m1, m2 consist of a letter and a space  Now (ASCII code for space) ⊕ m1 ⊕ m2 = (ASCII code for space) ⊕ 01101000  Let’s assume m1 = (ASCII code for space) then  (ASCII code for space) ⊕ m1 = 00000000 because XORing two identical values produces zero.  But 00000000 ⊕ m2 = m2 because XORing a given value with zero leaves the value unchanged.  Therefore, m2 = (ASCII code for space) ⊕ 01101000 = 00100000 ⊕ 01101000  m2 = 01001000 which is the letter ‘H’, i.e. the fourth character of m2 is ‘H’  Check:  key = 000100011111110110001001110110111001001110111101  m1 = c1 ⊕ key = 010000010101000101000001001000000100001101010011  m2 = c2 ⊕ key = 010100000101100101010100010010000100111101001110  Separating these into bytes  m1 = 01000001#01010001#01000001#00100000#01000011#01010011  m2 = 01010000#01011001#01010100#01001000#01001111#01001110  In character form  m1 = ‘AQA CS’  m2 = ‘PYTHON’ |

6 Fundamentals of computer systems

6.1 Hardware and software

Chapter 6.1 Hardware and software

|  |  |
| --- | --- |
| 1) | Hardware is the physical components of a computer system. |
| 2) | Software is the programs that execute on the hardware. |
| 3) | Software can be either application software or system software. Application software solve problems for users; system software controls the operation of the computer. |
| 4) | Application software can be either general purpose, specific purpose or bespoke. General purpose is suitable for many application areas; specific purpose is used for a particular application; bespoke software is tailor-made and has been written from scratch to solve a specific problem. |
| 5) | * Application software has to be translated from the language it was written in to a format that can be executed on the computer. * Application software needs to be loaded into memory. * It may need to obtain input from input devices, output data to output devices or communicate with other computers. * It may need to access data or store data onto secondary storage.   These operations are provided by the system software without which it would not be possible to run application software. |
| 6a) | To control the entire operation of a computer. |
| 6b) | A program designed to complete a common task. |
| 6c) | A collection of compiled routines that other programs can link to and use. |
| 6d) | Translate a program written in a computer language into a format executable on a computer system. |
| 7) | Anti-virus software, file compression programs, program to partition a hard disk. |
| 8) | There are two main roles: to hide the complexities of the hardware from the user and to manage hardware resources. |

6.2 Classification of programming languages

Chapter 6.2 Classification of programming languages

|  |  |
| --- | --- |
| 1) | Machine code is a language consisting of bit patterns / binary codes that a machine can interpret. |
| 2) | A machine code instruction is an operation that a machine is capable of carrying out. |
| 3) | Because there is a direct relationship with the hardware – a machine code instruction is an operation that the hardware can carry out. |
| 4) | Assembly language is the symbolic form of machine code; a name is assigned to each machine code instruction / operation. |
| 5) | There is a one-to-one relationship between machine code and assembly language. Each machine code instruction is equivalent to one assembly language instruction. |
| 6) | An assembler. |
| 7) | High level languages are closer to English than they are the machine. |
| 8) | There is a one-to-many relationship; a single high level language instruction is equivalent to many machine code instructions. |
| 9) | Programs written in an imperative high level language consist of a sequence of commands written by the programmer that solve a problem or accomplish a task. |
| 10) | Hand-coded assembly language when assembled can:  • achieve a smaller memory footprint  • achieve better code optimisation and therefore code will run faster  • directly access registers and low-level operating system routines. |
| 11) | Code written in assembly language or machine code is less readable than code written in a high-level language and so more difficult to understand, maintain and debug.  Code written in assembly language or machine code is less readable than code written in a high-level language and so more difficult to write without making errors  Code written in assembly language or machine code is machine dependent making it difficult to port to a different instruction set processor compared with code written using a high-level language which do port readily because they are not machine- oriented. |

6.3 Types of program translator

Chapter 6.3.1 Types of program translator

|  |  |
| --- | --- |
| 1a) | To translate assembly language into machine code. |
| 1b) | To read a program written in a high level programming language and translate it into an equivalent program in another language - the target language (often machine code). |
| 1c) | To execute a high level programming language program, statement by statement, by recognising the statement type of a statement, e.g. X = X + 1, and then calling a pre-written procedure/function for the statement type, to execute the statement. |
| 2) | Three possible answers:  An interpreter both “translates” and executes whereas a compiler only translates.  A compiler produces a separate independently executable form of the source code program whereas an interpreter does not.  A compiler is not needed when target form of source program is executed whereas in the case of the interpreter, execution requires the source code form of the program together with the interpreter, i.e. the interpreter needs to be available on the machine where the program is being run. |
| 3a) | When direct access to hardware is required, like processor registers or I/O controller registers. This is the case when writing device drivers, e.g. a screen driver.  For time-critical sections of code where execution speed is important, e.g. interrupt service routines, assembly language still has a role to play because in the hands of a skilled programmer, assembly language code can be written that is highly optimised for speed. |
| 3bi) | One-to-one (when translated an assembly language instruction will become a machine code instruction) |
| 3bii) | One-to-many (when translated a high-level language instruction will become multiple machine code instructions) |
| 4a) | When producing commercial software or where there is a requirement is to protect the algorithm or coding technique used because the source code will not be seen by the end-users.  When speed of execution is important as compiled code will execute faster than interpreting the source code.  When the interpreter may not be available on the machine where the program is to be executed. |
| 4b) | When developing programs as the interpreter can be helpful in the debugging process. |
| 5a) | Step 1  Use assembly language to write a new program – an interpreter that takes the intermediate code produced by HLL2 and can translate and execute it on the computer being used.  Step 2  Use the assembler to translate the assembly language program just written and convert it into an executable file (machine code).  Step 3  Run the executable file and HLL2 is now running on the computer and it can be given source code written in HLL and translate it into the intermediate code (which can then be interpreted using the interpreter that is now available on this computer). |
| 5b) | Step 1  HLL2 can be used to convert the source code for HLL1 into intermediate code.  Step 2  The interpreter can then be used to translate and execute the intermediate code. This means that HLL1 is now running on the computer. |
| 5c) | Once it is running (see part b), HLL1 can be given its own program source code to compile into machine code. |

6.4 Logic gates

Chapter 6.4.1 Logic gates

|  |  |
| --- | --- |
| 1) |  |
| 2) | |  |  |  |  | | --- | --- | --- | --- | | **X** | **Y** | **Z** | **Q** | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | |
| 3) | OR  AND  AND  X  Y  Z  Q |
| 4) | NOT  OR  AND  AND  X  Y  Z  Q |
| 5) | a) 1  b) 0 |
| 6) | a) 1  b) 0 |
| 7) | a) 0  b) 1  c) 1 |
| 8) | a) 1  b) 0  c) 0  d) 0 |
| 9) | It works out when the two 0 inputs (A0 and B0) are the same as each other and the two 1 inputs (A1 and B1) are also the same as each other. |
| 10) | |  |  |  |  |  | | --- | --- | --- | --- | --- | | A | B | C | D | Q | | 0 | 0 | 1 | 1 | 0 | | 0 | 1 | 1 | 0 | 1 | | 1 | 0 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 | 1 | |
| 11) | |  |  |  | | --- | --- | --- | | A | B | Q | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |
| 12) | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **A** | **B** | **C** | **D** | **Q** | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 1 | | 0 | 0 | 1 | 0 | 1 | | 0 | 0 | 1 | 1 | 0 | | 0 | 1 | 0 | 0 | 1 | | 0 | 1 | 0 | 1 | 0 | | 0 | 1 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | 1 | | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | 0 | | 1 | 0 | 1 | 1 | 1 | | 1 | 1 | 0 | 0 | 0 | | 1 | 1 | 0 | 1 | 1 | | 1 | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | 0 | |
| 13) | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 1 | 0 | 1 | |
| 14) | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 1 | 1 | | 1 | 1 | 0 | |
| 15) | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | A | B | D | C3 | C2 | C1 | C0 | | 0 | 0 | 1 | 0 | 0 | 0 | 1 | | 0 | 1 | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | 1 | 0 | 0 | | 1 | 1 | 1 | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | A | B | D | C3 | C2 | C1 | C0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | 1 | 0 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 16) | A.B + C.D = Q |
| 17) | = Q |
| 18) | Sum will be 0 (1 XOR 1)  Carry will be 1 (1 AND 1) |
| 19) | Sum will be 1  (A XOR B is 0; result of 0 XOR Carry in is 1)  Carry out will be 1  (A XOR B is 0; result of 0 AND Carry in is 0)  (A AND B is 1)  (0 OR 1 is 1) |
| 20) | 1) The enable clock signal input is set to 0.  2) Input D is set to 1.  3) The enable clock signal is set to 1.  4) Next time there is a rising edge on the clock signal the value of Q will be changed to match the value of the input D (1).  5) The enable clock signal input is set to 0. |
| 21) | a) Row 4  b) first Q is 1; second Q is 0 |

6.5 Boolean algebra

Chapter 6.5.1 Using Boolean algebra

|  |  |
| --- | --- |
| 1) |  |
| 2) |  |
| 3) | **Question 1**  The two shaded columns are the same.   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A | B |  |  |  |  |  |  |  |  | | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |   **Question 2**  The two shaded columns are the same.   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A | B | C |  |  |  |  |  | A.B.C | | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | |
| 4) |  |
| 5) |  |
| 6) |  |
| 7) |  |
|  | You were specifically asked to use Boolean algebra. The alternative approach is to use a truth table. Herewith the truth tables for Questions 4, 5, 6, 7.  **Question 4**  The two shaded columns are the same.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | A | B |  |  |  | | 0 | 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | 0 | | 1 | 1 | 0 | 0 | 0 |   **Question 5**  The two shaded columns are the same.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | A | B |  |  |  |  | | 0 | 0 | 1 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 | 1 | 1 |   **Question 6**  The two shaded columns are the same.   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A | B | C |  |  |  |  |  |  | | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |   **Question 7**  The two shaded columns are the same.   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A | B | C | D | A +B + C + D |  |  |  |  |  |  |  |  |  |  | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | |
| 8) |  |
| 9) |  |
| 10) |  |
| 11) |  |
| 12) |  |
| 13) |  |
| 14) |  |
| 15) |  |
| 16a) |  |
| 16b) |  |
| 16c) |  |
| 16d) |  |
| 16e) |  |
| 17) | W.X+W.Y.Z  Alternative answer: W.(X+Y.Z) |
| 18a) |  |
| 18b) |  |
| 19a) |  |
| 19b) |  |
| 20) | Alternative answer:  By applying identities and simplification laws there are also lots of other possible answers to either of the answers provided e.g. |
| 21) |  |
| 22) |  |

7 Fundamentals of computer organisation and architecture

7.1 Internal hardware components of a computer

Chapter 7.1.1 Internal hardware components of a computer

|  |  |
| --- | --- |
| 1a) | Processor, main memory, I/O controllers, buses |
| 1b) | Keyboard, printer, magnetic disk drive |
| 2) | A parallel bus consists of a set of parallel wires; each wire carries a single bit at a time. A serial bus carries data in a serial-bit form (one bit after another). |
| 3) | System bus. It is used to connect the main memory, processor and I/O controllers together. |
| 4) | Control bus, data bus, address bus |
| 5) | Only one component can successfully transmit at a time. |
| 6) | The purpose of the data bus is to provide a bi-directional path for moving data and instructions between components. The width of the bus is a key factor in system performance as a wider data bus means that more bits of data and information can be transmitted at a time so fewer main memory accesses are needed. This can lead to the processor taking less time to execute a program or process data. |
| 7) | The control bus is a bi-directional path used to send signals between components. The purpose of the control bus is to transmit command, timing and specific status information between components.  Command: Memory Write  Status: Transfer ACK  Timing: Clock |
| 8) | The maximum possible memory capacity of the system is 220 (1048576 memory locations). It also means that the address bus has a width of 20. |
| 9) | 216 (65536) |
| 10) | As a contiguous collection of memory block of read/write memory locations that are randomly accessible. Each memory location stores one word with each word consisting of the same number of bits. |
| 11) | Volatile means that the contents of each memory location are lost when the power is turned off. |
| 12) | Memory locations can be visited in any order (does not have to be contiguous). The time taken to access any memory location is the same as any other. |
| 13) | An I/O controller is a board of electronics that enables the processor to control and communicate with a peripheral device through an I/O port. |
| 14) | Each peripheral operates in a different way and it would not be sensible to design processors to directly control every possible peripheral. Otherwise, the invention of a new type of peripheral would require the processor to be redesigned. |
| 15) | An I/O port is a set of registers connected to the system bus in an I/O controller. An I/O port provides a standard interface through which a processor connected to the system bus can control and communicate with the peripheral device connected to the I/O controller. |
| 16) | In the Harvard architecture there are two main memories – one for instructions and one for data. In the von Neumann architecture the instructions and data are kept in the same memory. |
| 17) | The von Neumann architecture is typically used in general purpose machines; Harvard architecture when the work the processor undertakes is restricted to a specific task. |

7.2 The stored program concept

Chapter 7.2.1 The meaning of the stored program concept

|  |  |
| --- | --- |
| 1) | Both the program and the data on which it performs processing and calculations are stored in memory together. |
| 2) | The contents of the memory location are interpreted as an instruction when the next instruction pointer references it, and as data when an instruction references it. |
| 3a) | An advantage is that when a program needs to be downloaded from a remote location it can be treated as data and downloaded in the same way that an email would be. |
| 3b) | A disadvantage is that computer viruses are programs and get treated as data when being downloaded but as programs when the host computer is tricked into executing them. |

7.3 Structure and role of the processor and its components

Chapter 7.3.1 The processor and its components

|  |  |
| --- | --- |
| 1) | • Control Unit, which fetches instructions from memory, decodes and executes them one at a time.  • Arithmetic and Logic Unit (ALU) which performs arithmetic and logical operations on data supplied in registers, storing the result in a register.  • Dedicated registers: special purpose memory locations internal to the processor that support fast access as well as rapid manipulation of their contents.  • System clock, which generates a continuous sequence of clock pulses to step the control unit through its operation. |
| 2) | A processor interacts with this memory in 3 ways:  • it fetches instructions  • it loads a memory word into a processor register  • it changes the content of a memory word by a store operation. |
| 3) | The control unit sequences the operation of the processor by sending out control signals at the appropriate moments in time so that instructions may be fetched from memory in sequence, decoded and executed them one at a time. |
| 4) | The control unit fetches an instruction into the Current Instruction Register via the Memory Buffer Register and the data bus by  • reading the contents of the Program Counter to obtain the memory address of the memory word containing the instruction  • placing this memory address in the Memory Address Register connected to the address bus so that the addressed memory word can be selected and transferred across the data bus into the Memory Buffer Register  • transferring the instruction fetched from the memory from the Memory Buffer Register into the Current Instruction Register  The control unit then  • decodes the instruction to determine if it is one of a load, store, arithmetic operation, or logic operation  • executes the instruction by   * using the instruction’s operand fields as addresses to use in load or store operations, if required, or * loading a memory word into a register, or * changing a word of memory in a store operation, or * controlling an arithmetic operation or a logical operation in the Arithmetic and Logic Unit (ALU) using as operands the instruction’s operand fields. |
| 5) | The system clock provides regular clock pulses that the control unit uses to sequence its operations. |
| 6) | The number of clock cycles that fit one second. |
| 7) | Use one clock cycle to fetch an instruction from memory and another clock cycle to execute the instruction.  See Figure 7.3.1.3 for an example of a diagram that could be used to illustrate the answer to this question. |
| 8) | 2Ghz / 2 so 1,000,000,000 instructions per second. |
| 9) | Registers are memory locations internal to the processor that support fast access as well as rapid manipulation of their contents. |
| 10) | 1) There will be a much more limited number of registers than main memory locations.  2) Registers will have their own dedicated pathway within the processor whereas main memory locations have a shared pathway.  3) Access to main memory is much slower than access to registers.  4) Main memory is a cheaper technology than registers. |
| 11) | General purpose registers are registers that can be used by the programmer to store data, as needed. |
| 12) | Registers that are designed to be used by the control unit in a specific way. |
| 13a) | The Program Counter points to the next instruction to be fetched and executed. |
| 13b) | The MBR is connected to the data bus and contains a word to be stored in memory, or a word copied from memory. |
| 13c) | The MAR is connected to the address bus so that the memory address it contains can appear on this bus and be used at the memory end of this bus to select a particular memory word. |
| 13d) | The CIR contains the instruction fetched from memory while the control unit decodes and executes it. |
| 13e) | The status register stores single bit condition codes each of which indicates the outcome of arithmetic and logical operations carried out in the ALU. |
| 14) | Sign, zero, overflow and carry are four examples of condition codes. Supervisor and interrupt enable/disable are two examples of control flags. |
| 15) | The Arithmetic and Logic Unit (ALU) performs arithmetic and logical operations on data. |
| 16) | The Negative flag condition code (N) in the status register is set to 0 because the result is not negative. The Zero flag (Z) is set to 1, the Carry flag (C) and Overflow flag (O) are set to 0. The Interrupt Enable flag is 1 therefore enabling interrupts. The Supervisor mode flag (S) is 0, therefore the processor is in User mode.  The Arithmetic and Logic Unit (ALU) performs the arithmetic operations on the data supplied in the registers (4 and -4), storing the result in a register. |

Chapter 7.3.2 The Fetch-Execute cycle and the role of registers within it

|  |  |
| --- | --- |
| 1) | Program CounterMemory Address RegisterMemory Buffer RegisterCurrent Instruction Register |
| 2) | MAR 🡨 [PC]MBR 🡨 [Memory]addressed; PC 🡨 [PC] + 1CIR 🡨 [MBR][CIR] opcode part decoded and executedThe Program Counter points to the memory location containing the next instruction to execute. The contents of the Program Counter are copied to the Memory Address Register. This is then transferred via the address bus to the main memory and the contents of the specified memory location is then transferred, via the data bus, from the main memory to the Memory Buffer Register. At the same time as this the contents of the Program Counter are incremented so that the Program Counter continues to point at the next instruction to execute.The contents of the Memory Buffer Register are then copied to the Current Instruction Register. The contents (the instruction) is then decoded by the Control Unit and executed. During the execution stage the precise details of how the registers are used will depend on what the opcode of the instruction in the CIR is e.g. further data may need to be fetched from the main memory or for a branch instruction the contents of the Program Counter may need to be changed as the next instruction to execute is in a different part of the computer’s main memory. |

Chapter 7.3.3 The processor instruction set

|  |  |
| --- | --- |
| 1a) | 7 |
| 1b) | 1 |
| 1c) | 12 |
| 2a) | 28 (256) |
| 2b) | Because some of the possible bit patterns may not map onto any defined machine operations – there may be more possible bit patterns in the number of bits reserved for the opcode than there are defined machine operations. |
| 2c) | The basic machine operation and the addressing mode. |
| 3) | The set of bit patterns for which machine operations have been defined. |
| 4) | Three examples that could be used in the explanation:  000000 0000 0001 1101  LDR R0 29 - is an example of an instruction that has two operands (one a register and one a memory location); it loads the contents of memory location 29 into the register R0.  000010 0001 0001 1111  STR R1 31 - is an example of an instruction that has two operands (one a register and one a memory location); it stores the contents of register R1 in the memory location 31.  000100 0001 0011 0111  ADD R1 R3 7 - is an example of an instruction that has three operands (two registers and one numeric value); it adds the number 7 to the contents of register R3 and stores the result in register R1.  000101 0001 0011 0111  ADD R1 R3 R7 - is an example of an instruction that has three operands (three registers); it adds the contents of register R7 to the contents of register R3 and stores the result in register R1. |

Chapter 7.3.4 Addressing modes

|  |  |
| --- | --- |
| 1a) | 84 |
| 1b) | 140 |
| 2) | 77 |
| 3) | 0xFCC1 will contain 0  0xFCC2 will contain 65535 |
| 4) | 4D + FCC0  In decimal this is 77 + 64,704  So contents will be 64,781 |

Chapter 7.3.5 Machine-code/assembly language operations

|  |  |
| --- | --- |
| 1a) | 21 |
| 1b) | 42 |
| 1c) | 84 |
| 2) | 67 |
| 3a) | 78 |
| 3b) | 25 |
| 4) | One possible answer:  MOV R1, #5  MOV R2, #3  MOV R3, #6  ADD R2, R3, R2  ADD R0, R1, R2 |
| 5) | MOV R1, #5  MOV R2, #13  SUB R0, R2, R1 |
| 6a) | Negative |
| 6b) | Zero |
| 6c) | No condition flags are set. |
| 7a) | Negative |
| 7b) | Zero |
| 7c) | No condition flags are set. |
| 8a) | N = 1  Z = 0  C = 0  O = 0 |
| 8b) | N = 0  Z = 0  C = 0  O = 0 |
| 8c) | N = 0  Z = 1  C = 1  O = 0 |
| 9) | After storing initial values in R0 and R1, it keeps adding 1 to the contents of R1 until it contains the same value as R0. |
| 10a) | After storing initial values in R0 and R1, it keeps subtracting 1 from the contents of R0 until it is no longer greater than the contents of R1. |
| 10b) | After storing initial values in R0 and R1, it keeps adding 1 to the contents of R1 until it is no longer less than the contents of R0. |
| 11) | BNE |
| 12a) | 0 |
| 12b) | 7 |
| 12c) | 7 |
| 13) | AND R1, R0, #1  CMP R1, #0x15  BEQ StartProcess |
| 14) | ORR R0, R0, #8 |
| 15) | AND R1, R0, #1  AND R2, R0, #4  The first instruction will isolate bit 1 and store it in R1; the second instruction will isolate bit 3 and store it in R2. The contents of register R1 can be compared with 0, if it is 0 then bit 1 was 0 (else it was 1). The contents of register R2 can be compared with 0, if it is 1 then bit 3 was 0 (else it was 1).  Alternatively,  AND R2, R0, #5  This will isolate bit bit 1 and bit 3. |
| 16) | Logical OR can be used to make bits be set to 1 (irrespective of what their value was before) e.g. ORR R0, R0, #1 will cause bit 1 of register R0 to be set to 1. ORR R0, R0, #4 will cause bit 3 of register R0 to be set to 1. ORR R0, R0, #5 will cause bits 1 and 3 of register R0 to be set to 1. The reason for this is when you OR a bit with a 1 the result will always be a 1.  Logical AND can be used to make bits be set to 0 (irrespective of what their value was before. The reason for this is that if you AND a bit with the value 0 it will always become 0. So, AND R0, R0, #1 will cause all but the first bit of R1 to become 0. AND R0, R0, #4 will cause all but the 3rd bit of R1 to become 0. AND R0, R0, #5 will cause all but the 1st and 3rd bits to become 0. |
| 17) | To invert:  EOR R0, R0 #0xFFFFFFFF  To restore the same command could be used. |
| 18) | LSL R0, R0, #0x8 |
| 19) | LSL R0, R0, #0x4 |
| 20) | After MOV instruction R0 is:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |   After LSL instruction R0 is:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |   and the carry bit will be 1. |
| 21) | #0x00000001 |
| 22) | LSR R1, R0, #0x10 |
| 23) | After MOV instruction R0 is:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |   After LSL instruction R0 is:   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |   and the carry bit will be 1. |
| 24) | LDR R0, #0x1000  ADD R0, R0, #100  STR R0, #0x1000 |
| 25) | LDR R0, #0x1000  CMP R0, #5  BGT addone  SUB R0, R0, #1  B end  addone: ADD R0, R0, #1  end: STR R0, #0x1000 |
| 26) | LDR R0, #0x1000  loopstart: CMP R0, #9  BGT end  ADD R0, R0, #1  B loopstart  end: STR R0, #0x1000 |
| 27) | LDR R0, #0x1000  loopstart: SUB R0, R0, #1  CMP R0, #0  BNE loopstart  STR R0, #0x1000 |
| 28) | LDR R0, #0x1000  LSL R0, R0, #3  STR R0, #0x1000 |
| 29) | LDR R0, #0x1000  AND R1, R0, #4  CMP R1, #1  BNE end  MOV R2, #0  STR R2, #0x1004  end: |
| 30) | Depending on the simulator used the command HALT may need to be added at the end of each of the assembly language programs. |

Chapter 7.3.6 Interrupts

|  |  |
| --- | --- |
| 1) | An interrupt is a signal from some device/source seeking the attention of the processor. |
| 2) | Power failure, arithmetic overflow, generated by an I/O controller |
| 3) | Interrupts provide a mechanism by which a program currently executing on the processor may be interrupted so that another program may be run if it is of higher priority, or so that a temporarily suspended program’s status can be set to runnable because the resource it needs in order to run has become available or so that the processor may be shared amongst more than one program in turn, on a rotating priority basis, with each turn consuming a small slice of time controlled by a timer interrupt. |
| 4) | An interrupt service routine is a small piece of program code written to process an event such as a key on a keyboard being pressed. |
| 5) | A typical sequence of actions when an interrupt occurs would be:   1. The processor must complete the current fetch-execute cycle for the current program if begun. 2. The contents of the program counter, which points to the next instruction of the current program to be executed, must be stored away safely so it can be restored after servicing the interrupt. 3. The contents of other registers used by the current program are stored away safely for later restoration. 4. The source of the interrupt is identified. 5. Interrupts of a lower priority are disabled. 6. The program counter is loaded with the start address of the relevant interrupt service routine. 7. The interrupt service routine is executed. |
| 6) | The volatile environment of the processor refers to the contents of processor registers, e.g. the program counter, the general purpose registers, the status register. |
| 7) | The volatile environment needs to be saved, so that it is possible to restore the contents of the registers to the state they were in before the interrupt occurred and the interrupted program can continue from the exact point at which it was interrupted. |

Chapter 7.3.7 Factors affecting processor performance

|  |  |
| --- | --- |
| 1) | For Processor 1 the average number of instructions executed per second is 1/3 \* 5 => 1.7 billion.  For Processor 2 the average number of instructions executed per second is 2/3 \* 3 => 2 billion.  Therefore, processor 2 was faster at executing these programs. |
| 2) | The number of instructions in the program, the clock cycle time and the average CPI. |
| 3) | Multiple cores means multiple ALUs which means that arithmetic and logical instructions can be executed in a reduced amount of time as a single instruction can use all four ALUs at the same time. |
| 4a) | Increasing the width of the data bus means that more bits can be transferred at a time meaning that fewer transfers are needed. Doubling the data bus means that double the number of bits can be transferred at the same time. |
| 4b) | This is approximately a 20% increase in memory clock speed. Increasing the memory clock speed will mean more clock cycles and therefore a faster rate of data transfer. |
| 5) | There is also latency as the memory doesn’t respond immediately to a read or write request. |
| 6) | Cache memory is much quicker to access than the main memory. It is used to store data from frequently used memory locations, pre-fetched instructions and data to be written to the main memory. Operations using cache memory can be completed quicker than those using the main memory. |
| 7) | The word length is the number of bits that a processor can manipulate at one time. If a processor needs to work with more bits than are allowed by the word length then it will have to execute more instructions to be able to do so. More instructions results in a longer execution time. |
| 8) | A wider address bus means that a larger range of memory locations can be accessed. |

7.4 External hardware devices

Chapter 7.4.1 Input and output devices

|  |  |
| --- | --- |
| 1) | A barcode is a sequence of black and white bars that encode information. |
| 2) | The scanner shines a light onto the barcode and has photodiodes (sensors) that detect the amount of light reflected back. White bands reflect more than black bands. The pattern of reflection is converted from optical form to electrical form; this electrical form is then analysed and converted into characters. |
| 3) | So that if the barcode cannot be scanned the value encoded by the barcode can be entered manually e.g. using a keyboard. |
| 4) | The luggage will be moving at a fairly high speed when on the conveyor belts and barcode scanners can work reliably even when the luggage is moving. Also, the system is fairly cheap which is useful as lots of printed barcodes and barcode scanners are needed at a large airport. |
| 5) | Light reflected by objects is focused by the lens of the digital camera onto a two-dimensional array of photosensors (light-sensitive cells). Each cell accumulates an electrical charge proportional to the brightness of the illumination.  The magnitude of the charge in each cell is sampled and converted into digital format by an Analogue to Digital Converter (ADC).  Each cell is covered by one red, one blue and two green filters. |
| 6) | CCD would be used in professional photography as the quality of the final images is important (no one would buy grainy images). CMOS might be used in delivery drones which need a camera to use for image recognition - there is a limited amount of power available (CCD devices would drain the battery more meaning the drone’s delivery range would be more limited). |
| 7) | Because it prints a whole page at a time. |
| 8) | A page description language is used to describe the page to be printed (using lines, arcs, etc…).  A processor in the laser printer converts the page description into a bitmap image. A negative charge is applied to a photosensitive drum. Lasers are directed onto the drum, turned on and off according to the bitmap image. The lasers reverse the charge on the drum. The resulting pattern of charges on the drum’s surface is an image of the page printed.  The drum is exposed to negatively-charged black toner powder which will attach to the positively charged parts of the drum’s surface. The rotating drum is then rolled over the paper which transfers the toner to the paper. The paper is then passed through heated rollers which fuse the toner to the powder. |
| 9) | Four toners are used in a colour laser – Cyan, Magenta, Yellow and Key (Black). All the other colours can be obtained by mixing different proportions of the Cyan, Magenta and Yellow toners. A darker colour black can be obtained by using the Key (Black) toner than that obtained by mixing the other toners. |
| 10) | RFID is a method of transmitting data using radio frequencies. |
| 11) | An RFID tag consists of a microchip attached to an antenna which is packaged in a way that allows it to be applied to an object such as an item of clothing for sale. The tag picks up signals from and sends signals to a reader. |
| 12) | 1) EAS (Electronic Article Surveillance) systems that are used to detect the theft of goods from a shop.  2) Oyster cards are used in the London area as tickets for journeys on public transport.  3) RFID tags are often used to track the location of stock in shops. |
| 13) | RFID can be used even when items are in boxes / not visible whereas barcode scanning only works if the barcode is visible.  RFID can read data from multiple tags simultaneously whereas barcode scanners can only read one barcode at a time. |

Chapter 7.4.2 Secondary storage devices

|  |  |
| --- | --- |
| 1) | To retain a program we have created in RAM, or some information we have written to RAM. Without this the programs and data would not be retained when the computer is turned off. |
| 2) | Because when data is transferred between the computer and the hard disk one block of data is transferred at a time. |
| 3a) | One of the concentric rings on a platter of a hard disk. |
| 3b) | A subdivision of a track. |
| 3c) | The smallest unit of transfer between a computer and a disk. A disk block is one sector of one track. |
| 4) | The surface of a disk is covered in magnetized particles. Each particle can be aligned in one of two ways – equivalent to the binary values of 0 and 1. The particles are arranged in tracks. Tracks are divided into sectors.  When in use the disk spins round. There is a read/write head that moves radially until it is over the desired track of the spinning disk. When the correct sector of the track is underneath the read/write head the data block is read from/written to that sector. |
| 5) | Overall, this will reduce latency. Smaller platters will reduce seek time as the radial movement for the read/write heads will be less. Faster rotational speed and smaller platters will both reduce rotational delay as the desired part of the track will be underneath the red/write head sooner. |

|  |  |
| --- | --- |
| 6) | An optical disk is a flat, usually circular disc which encodes binary data (bits) in the form of pits (binary value of 0 due to lack of reflection when read) and lands (binary value of 1 due to a reflection when read) on a reflective material, usually metallic, on one of its flat surfaces. |
| 7) | The data is written on the discs using disc-mastering machinery that impresses pits (physical depressions) into a continuous spiral track. The silvery data surface contains pits in a single track 3.5 miles (5.6 km) long.  The disc spins at 200-500 revolutions per minute depending on which part of the track is being read. Data is read by shinning a laser beam onto the reflective metal layer where the pits are impressed.  More laser light is reflected from the unpitted surface than from the pitted surface – this corresponds with the binary values 1 and 0. The amount of reflected light is detected by a sensor and converted into a digital form. |
| 8) | SSDs operate by trapping electrons in a wafer of semiconducting material. These electrons and their electric charge remain trapped even when electric power is removed. Binary 0 is represented by trapped electrons and binary 1 by absence of trapped electrons. The sites (floating gate transistors) where these electrons are trapped are organized in a grid. The entire grid layout is referred to as a block, while the individual rows that make up the grid are called a page.  The technology used is NAND flash memory. A solid-state disk is a block-oriented storage device which has to erase a block first in order to rewrite it because unlike magnetic hard disk drives, NAND flash memory can’t overwrite existing data. Erasing a block in the SDD means “untrapping” electrons.  The solid-state disk drive has an onboard controller which consists of an embedded microprocessor with RAM buffer to perform reading and writing to the solid state disk. To alter the contents of a particular memory location of SSD storage, an entire block must be constructed containing the new information and written to SSD. The controller arranges for this new block to be written to a different area of SSD. The reason for this is that SSD blocks can be programmed for only a limited amount of time before they become unreliable. To lessen this effect, a controller uses a technique called wear-levelling, which effectively makes sure that all the drive’s memory chips are used, cell by cell, before the first cell will be written on again. |
| 9) | Four possible answers:   * Lower power consumption * Faster boot speed for operating systems * More robust (no moving parts) * Generate less heat |
| 10a) | Online storage of programs and data. |
| 10b) | Distributing software. |
| 10c) | Distributing software or videos. |
| 10d) | Backing-up data. |
| 10e) | Distributing high-definition video. |
| 11a) | Dual-layer DVD-R |
| 11b) | Dual-layer DVD-RAM |

8 Consequences of uses of computing

8.1 Individual (moral), social (ethical), legal and cultural issues and opportunities

Chapter 8.1 Individual (moral), social (ethical), legal and cultural issues and opportunities

|  |  |
| --- | --- |
| 1) | Personalised search results are when the results returned by a search are determined not just on the keywords entered but also by personal data about the user e.g. other websites the user has visited through previous search results. |
| 2) | Personalised adverts are adverts that are displayed to the user based on personal data that is known about them i.e. the same webpage will have different adverts displayed alongside it for different people; the adverts displayed will depend on the personal data recorded about that user. An example of this is that when searching for cricket an entomologist might get shown adverts about the latest books about insects whereas a cricketer might get shown adverts for new sporting equipment. |
| 3) | It makes it less likely that people will encounter opinions different to their own potentially resulting in a less tolerant society.  The company providing the search engine are making judgements about what people should / should not see and the people who wrote the algorithms making these judgements may project their own biases onto the results. |
| 4) | Two possible arguments for and against are given, but there are many other possible correct answers.   * Personalising search results makes it more likely that the information a user is searching for will be returned in the first pages of suggested links saving time for the user. * On social media sites it is more likely that you will be shown adverts for products/services of interest to you and you may not have found out about these products/services without the user of this personalization. * The search engine / social media sites may sell personal data to other organisations that you would prefer not to have personal data about you. * The search engine / social media sites may be able to derive knowledge about you that you would prefer to be kept private. |
| 5) | The impact of a piece of software may provide both benefits and drawbacks for society. An example is Tor which has allowed journalists to communicate securely with whisteblowers and dissidents without allowing others to track them down. However, the Tor software has also enabled criminals to hide their activity more successfully from law enforcement agencies. |
| 6) | Software is designed which means that the moral and cultural values of the designers are likely to be embedded in the software.  The examples given below are just a sample of ways that software is designed with embedded moral and cultural values of the designers. Some of the examples can be used in more than one category but have not been for brevity’s sake.  **Privacy:** Privacy is the freedom granted to individuals to control their exposure to others. It is a fundamental right. Any example which promotes or demotes privacy, e.g. Google Street View overrides people’s right to privacy, <http://www.theguardian.com/world/2009/nov/29/google-street-view-south-africa>  The use of software that contains spyware or otherwise leaks personal data to third parties.  **Freedom from bias:** Software that discriminates against specific groups of users by the information content that is returned from a search on the basis of ethnicity, location (social), income group, etc; the way that information is communicated, e.g. only for sighted people; placing certain search results before others to favour certain organisations or opinions/views of the world (cultural); assumptions made about the user’s physical abilities mean that the software can only be used effectively by certain people, e.g. the able-bodied; game software that is not gender neutral but targets one gender group over another; educational software that assumes particular cultural norms; loan approval software that discriminates unfairly against loan applicants with ethnic sounding names.  **Universal usability:** Software designers make assumptions in their user interface design that restricts use of their software to specific groups of people whilst discriminating against other groups, e.g. user actions require users to be able to read text. This discriminates against people who are unable to read the text because of a physical disability or because of illiteracy; requires a fast Internet connection to use the system therefore discriminating against groups of users for which fast Internet access is unaffordable or physically impossible.  **Trust:** Safety critical software designed in such a way that it is vulnerable to hacking, e.g. driverless car, heart pacemakers. Sensitive medical records with insufficient protection against hacking. Software designed at behest of governments to allow monitoring of its use, e.g. user trusts software to be secure against monitoring of encrypted communications but the software designers have built in a backdoor to enable government agents to spy on the user’s communication.  **Autonomy:** Refers to the control that a user has over their own activities/work and the means used to realise personal goals.  Email promotes communication and in the work place this can lead to more sharing of ideas/information and across hierarchies which in turn offer workers better opportunities to control aspects of their work and express their views; remote working/telecommuting made possible and thus people given greater autonomy over where they work, its physical setup, the pace of work as well as being freed from being constrained to office hours and direct supervision by management and co-workers or one place. Better informed workers because of increased and better access to information, e.g. intranets and the World Wide Web, leads to more job autonomy and independence.  However, user autonomy is undermined by third party monitoring software. This can lead to users feeling that they are being constantly surveilled with the consequence that users’ behaviour becomes that which they think their observers expect thus dehumanising them and making them feel more like machines.  **Informed consent:**  Software supports experiments on its users without users being given the option to consent, e.g. Facebook experiment –  <https://pando.com/2014/07/23/these-researchers-are-studying-facebooks-news-feed-to-make-algorithms-more-accountable/>  <http://searchdatamanagement.techtarget.com/opinion/Facebook-experiment-points-to-data-ethics-hurdles-in-digital-research>  **Human welfare:** Safety critical software designed in such a way that it is vulnerable to hacking, e.g. driverless car, heart pacemakers.  **Identity:** Software is designed in a such a way that it offers insufficent protection against identity theft, e.g. some airline boarding passes contain information that can be linked to user accounts in sufficent detail to steal user identities for fraudulent purposes.  **Democracy/freedom of information:**  Filtering software enables access to information to be controlled, e.g. China’s censoring of access to information on the World Wide Web via control of search engines, social media sites, etc  Other categories to explore how software embeds moral and cultural values:  **Ownership and property:**  **Environmental sustainability:**  **Justice:** |
| 7) | Laws are generally local to one country and the Internet is global. Laws made in one country may not be able to be applied successfully across the Internet that embodies a multitude of different cultural, ethical and legal systems from many countries. |
| 8) | Computers do not only carry out actions using information in the form of programmed instructions but they can also produce information that coded algorithms can capture and report. However, whilst you have some control over the former in that you have initiated the action, e.g. a search, you seemingly have much less control over the latter. This is what is meant by asymmetry of power.  Pervasive surveillance is the capturing of small data from individuals’ online actions and utterances as they go about their daily lives. |
| 9) | An Internet of Things could soon be widely used within our own houses e.g. our fridge could automatically place an order for milk to be delivered when it is running low. Another example is insurance firms can install machines in cars that monitor driving habits (often in return for discounted insurance).  Precautions that should be taken by the designers of such systems include security of data e.g. if milk has not been ordered for a long time this may suggest the house is unoccupied and if this data is not secure your house could be at risk of burglary; if your car’s location is known then this information could also be misused in the same way.  Another precaution that should be taken by the designers is the right to privacy. The collection of data via machine to machine systems could result in personal knowledge being derived which individuals may prefer to keep private. If your car’s location is frequently logged then algorithms may be able to derive is you are suffering from health problems (e.g. your car is often used to drive to a specialized medical facility); if medicines are kept in your fridge then a similar privacy infringement could occur. |

9 Fundamentals of communication and networking

9.1 Communication

Chapter 9.1.1 Communication methods

|  |  |
| --- | --- |
| 1) | Serial transmission is when bits are sent one after another along a single wire. |
| 2) | Parallel transmission is when bits are sent along several wires simultaneously. |
| 3) | Two disadvantages of parallel data transmission are that it has a lower data transmission rate at which it can be reliably operated at and a shorter distance over which it can be reliably sent. |
| 4a) | Synchronous transmission is when the communicating endpoints’ interfaces are continuously synchronized by a common clock. |
| 4b) | Asynchronous means that data is transferred without support from an external clock signal. |
| 5) | Asynchronous transmission is relatively cheap as less hardware is needed. It is normally the most appropriate choice when messages are generated at irregular intervals.  However, there is a higer proportion of non-data bits sent in asynchronous transmission as control (stop and start) bits are used. |
| 6) | The start bit is used to wake up the receiver. The receiver’s clock is set ticking by the start bit. This enables the receiver to sample the data correctly by generating clock pulses synchronised to the bits in the received data. Effectively, transmitter and receiver clocks are synchronised by the arrival of a start bit.  The time interval for the stop bit allows the receiver to deal with the received bits, i.e. transfer them into the RAM of the computer, before receiving and processing the next serial frame. |

Chapter 9.1.2 Communication basics

|  |  |
| --- | --- |
| 1) | Every hundredth of a second. |
| 2) | 1000 |
| 3a) | Number of signal (voltage) changes per second. |
| 3b) | Number of bits that can be transmitted per second. |
| 4) | The bit rate is equal to the baud rate multiplied by the number of bits that can be sent per signal. The bit rate can be higher than the baud rate if more than two voltage levels are used. |
| 5) | 3 |
| 6) | -6 000  -4.5 001  -3 010  -1.5 011  1.5 100  3 101  4.5 110  6 111 |
| 7) | 900 x 3 = 2700bps |
| 8) | Bandwidth is a measure of how fast a transmission media is. It is the range of signal frequencies that can be transmitted from one end of a communication link to the other without significant degradation in the signal strength. |
| 9) | Latency is the time delay from an action being initiated to its first effect begins. |
| 10) | A communication protocol is a set of pre-agreed signals, codes and rules used to ensure successful communication. |
| 11) | The higher the bandwidth the higher the bit rate that can be achieved. |

9.2 Networking

Chapter 9.2.1 Network topology

|  |  |
| --- | --- |
| 1) | Topology is the layout of the network. |
| 2) | See Figure 9.2.1.1a for an example diagram. |
| 3) | See Figure 9.2.1.2 for an example diagram. |
| 4) | This can be done using a switched Ethernet set up. Each device is connected to a central switch by its own link so is physically set-up as a star topology. However, it is logically a bus network as it still uses a bus network protocol. |
| 5) | A network adapter is the interface between a computer and a network cable. It performs all the functions needed for a computer to communicate on a network – converting data between the form it is stored in a computer and the form it is transmitted in along the cable. |
| 6) | A MAC address is a unique 48-bit value assigned to each network adapter so that it can be identified. |
| 7) | In a physical bus network a shared media is used for transmission (in baseband mode) meaning that if two adapters try to transmit a frame at the same time a collision will occur. Frequent collisions will have a significant impact on the speed of a network. |
| 8) | In switched Ethernet each device has its own two-way link to a central switch. When frames are received at the switch they are buffered and sent in-turn to the relevant device when the backbone is available. |
| 9) | Possible discussion points when comparing to a physical bus network:   * No collisions can occur (resulting in faster network operation) * Better security (each device has its own link to the switch so eavesdropping more difficult) * Break in the main cable for a physical bus network will have an effect the whole network   Possible discussion point when comparing to a star network:   * Can be slower under heavy traffic as buffer can overflow |

Chapter 9.2.2 Types of networking between hosts

|  |  |
| --- | --- |
| 1) | A network in which there is minimal or no reliance on dedicated servers. All computers are equal, and are called peers. Each peer may act as both a client and server. |
| 2) | A peer-to-peer local area network (LAN) is a good choice for environments where:   * there are fewer than 10 users   the users are all located in the same area and the computers will be located at user desks  security is not a major concern  the organisation and the network will have limited growth over the foreseeable future. |
| 3) | A server-based local area network is a client- server network in which resources, security, administration and other functions are provided by dedicated servers. Clients request services that are satisfied by dedicated servers.  It is considered to be a client-server network as some machines provide a service (servers) which other machines can request to use (clients). |
| 4) | In a client-server architecture, there is an always-on host, called the server, which services requests from many other hosts, called clients.  A situation where this would be used is for a server hosting a webpage – the server needs to be able to deal with requests from multiple client machines and it needs to be always-on as it is not known in advance when requests will be received. |
| 5) | In a peer-to-peer architecture, client hosts not only download a file, but can upload what they have obtained to others as well. Client hosts are capable of doing both at the same time. They are then called peers rather than clients.  This architecture is often used for file-sharing applications as it doesn’t result in servers being overloaded. |

Chapter 9.2.3 Wireless networking

|  |  |
| --- | --- |
| 1) | The purpose of WiFi is to provide a wireless connection between computing devices and to enable these devices to connect to the Internet. |
| 2) | Radio waves. |
| 3) | 1. Stations: computing devices with wireless network interfaces 2. Access Points (APs): provide the wireless-to-wired bridging function in which wireless frames are converted to wired frames; access points also control access to a wireless network by authenticating users that wish to join the network 3. Wireless medium: to move frames from station to station 4. Distributing system: used to connect several access points to form a large coverage area of the same LAN |
| 4) | An SSID is a user-friendly name for a network that is used to identify a Basic Service Set to users of a wireless network. |
| 5) | Channel selection is when a station selects the channel being used by the access point. A channel is a fixed-width frequency range from the frequency band. |
| 6) | A collision occurs when two transmitters are within interference range of each other, and they send at similar times. What actually collides are frames, one from each transmitter. The result is that neither frame will be properly decoded at the receiver. |
| 7) | The WiFi channel through which WiFi signals travel is a shared medium. |
| 8) | Each station tries to sense the presence of others on the shared medium and tries to avoid collisions by “listening” and not sending when another station is using the medium. The length of time it listens for before sending is called the wait and listen period and will be longer than the potential gap between a device receiving and decoding a transmission and then sending its ACK signal.  A station waits to send until the channel is clear. When a station has successfully received and decoded a transmission it sends an ACK signal to the sending station. If a sending station does not receive an ACK signal then it knows that a collision has occurred.  When a collision occurs the sending devices choose a random time slot from the contention window and resend after the wait and listen period + the number of time slots chosen (assuming no other transmissions from other devices occur during this time).  To prevent collision RTS (request to send) and CTS (clear to send) are used. When a device wishes to send it transmits a RTS signal. The receiver then sends a CTS signal. All devices within range of these signals will not try to transmit until the ACK signal has been received. |
| 9) | When two stations are not in range of each other – they won’t sense each other’s transmissions and so may try to transmit when the other is already doing so. This will cause a collision if one of the receiving devices is in the range of both the sending stations. |
| 10) | To prevent collisions. |
| 11) | When many collisions are occurring due to hidden stations. |
| 12) | They are security protocols used to secure wireless networks. |
| 13) | To prevent spoofing – when messages purport to come from a genuine user when they don’t.  To keep messages confidential (between the sender and intended recipient) by securing against unauthorised access.  Ensure the integrity of messages by detecting unauthorised alteration of data being transmitted. |
| 14) | To join a WPA/WPA2 secured wireless network there is an authentication stage that a client has to pass through which checks that the client knows a pre-shared secret key (PSK). This check is based on a message authentication code generated using the PSK. The access point is responsible for completing the authentication process and only clients that pass the authentication process are able to join the wireless network. |
| 15a) | In MAC address white list filtering, the access point has an internal table of MAC addresses which it consults to decide whether to permit access to the network or not. |
| 15b) | Wireless stations require a knowledge of the SSID in order to join the network. If broadcast, the SSID appears in the network settings window of stations within range. In this form of protection, an access point disables broadcasting its SSID to wireless stations. Thus, only clients who already know the pre- configured SSID can establish a connection. |
| 16) | MAC address white list filtering is not that effective as it is possible for a snooper to find out the MAC address of a station. If this is done then the operating system can change the information about MAC addresses in the packets that it sends out thus making it look like the packets come from an authorised device.  Disabling broadcasting of SSID is not that effective as it is possible for a snooper to find out the SSID. The reason for this is that the SSID will be included in unencrypted form in packets of data sent when authenticating a station trying to access the wireless network. |

9.3 The Internet

Chapter 9.3.1 The Internet and how it works

|  |  |
| --- | --- |
| 1) | Two or more networks connected together. |
| 2) | A network of computer networks, computers and devices with computing capability using globally unique IP addresses and TCP/IP. |
| 3) | Messages to be sent are split into a number of segments called packets. The packets of a message are allowed to travel along independent paths through a network of routers. Routers use a packet’s destination IP address to route the packet, taking account of how congested are particular routes are.  Routers are special packet switches that receive incoming packets of data along one link and send them as outgoing packets on another link. |
| 4) | The end-to-end principle states that the two endpoint hosts should be in control of the communication. The role of the packet switched network is to move packets between these two endpoints. |
| 5) | Possible advantages:  • The sending application in Computer X and the receiving application in Computer Y are able to survive a partial network failure.  • Packets can be rerouted around failures very quickly and sent along alternative paths.  • The internet can grow easily because control resides in the endpoints not in the internet.  • There is no requirement for routers to notify each other as endpoint connections are formed or dropped; this simplifies the design of routers.  • The integrity and security of each packet sent is handled by the endpoints (end-systems), which simplifies the role of the internet.  • Each endpoint need only be aware of the router to which it is directly connected and, optionally, a name resolution service that converts user-friendly hostnames into their corresponding IP addresses. |
| 6) | Routers are used because it is not practical to connect every host directly to every other host. |
| 7) | A router is a device that receives packets from one host or router and uses the destination IP address that they contain to pass on the packets, correctly formatted, to another host or router. |
| 8) | Routers are organised in a hierarchical structure on the Internet. |
| 9) | Gateways allow two or more networks that use different data link or network protocols to be connected so that information can be passed from one system to another. |
| 10) | Main components are: hardware destination address (MAC address), hardware source address (MAC address), source IP address, destination IP address, source port number, destination port number, sequence number, data (payload). |
| 11) | Routing uses source and destination IP addresses at the network layer level, and source and destination hardware addresses at the data link layer. |
| 12) | IP addresses remain unchanged; hardware addresses e.g. MAC addresses change at each stage/hop. |
| 13) | The IP addresses don’t change as these indicate the endpoints and the endpoints are not changed. The hardware destination addresses change as these are used to indicate the next stage/hop in the routing process i.e. the address of the next hardware device that the packet needs to be sent to and the address of the hardware device that is forwarding the packet – these two devices change at each stage/hop in the routing process. |
| 14) | A uniform resource locator (URL) is a short string that represents the target of a hyperlink. |
| 15) | Example URL: http://www.bbc.co.uk/sport/football/teams/ipswich-town  The how: http  The where: www.bbc.co.uk  The what: sport/football/teams/ipswich-town |
| 16) | The Domain Name System was invented so that people could use a memorable name to refer to a host on a network instead of an IP address. FQDNs are much easier for people to remember than the equivalent IP address. |
| 17) | It is organised as a hierarchical system of names and abbreviations. |
| 18) | Top level domains: org and com  Second level domains: ac and sch |
| 19) | Com – commercial  Org – non-commercial  Ac – academic  Sch – school |
| 20) | The purpose is to translate FQDNs into IP addresses. |
| 21) | In addition to translating FQDNs into IP addresses they also proved the following services:   * Host aliasing * Mail server aliasing * Load distribution |
| 22) | Internet registries store registered domain names together with their corresponding Internet addresses. |
| 23) | They would inform the other ISPs that they are connected to so that their routers are aware of the network ID and prefix that you have been allocated. |

Chapter 9.3.2 Internet security

|  |  |
| --- | --- |
| 1) | Packet filtering is done by a packet filter acting on a network-layer datagram. It allows packets through based on their IP address, or other datagram content. Information about packets is not remembered by the firewall. This type of firewall can be tricked very easily by hackers because allow/deny decisions are taken on a packet by packet basis and these are not related to the previous allowed/denied packets.  Stateful inspection is different from stateless packet filtering in that it rejects packets that a table of current outgoing TCP/UDP connections is maintained and that if a packet is received that does not belong to one of these connections it is rejected.  A firewall proxy server works at the application level of the TCP protocol stack and therefore filters by application protocol type, e.g. SSH, HTTP, and can provide authorisation by user controlling what content different users may access. |
| 2) | (a) If encrypted with X’s private key then it will be possible for people other than Y to decrypt the encrypted message since X’s public key is publicly available.  (b) If encrypted with X’s public key then it will not be possible for Y to decrypt the encrypted message since X’s private key is required for this and X’s private key is known only to X. |
| 3) | One possible method is to use asymmetric encryption to encrypt a symmetric key which is then sent. Both devices will then have a copy of the symmetric key which they can use to encrypt and decrypt each other’s messages.  Alternatively each device can generate a public key from their own private key. The public keys are then exchanged. Both devices can then calculate a shared key from the combination of their private key and the two public keys. |
| 4a) | Digital certificates are files containing information about the owner of a website, and the public half of an asymmetric key pair (e.g. RSA). A certificate authority (CA) digitally signs the certificate to verify that the information in the certificate is correct. By trusting the certificate, you are trusting that the certificate authority has done its due diligence.  A website that supports HTTPS should have a certificate and a corresponding public key. This will enable a connection to be made between a web browser and the website using the Transport Layer Security (TLS) protocol. |
| 4b) | The web browser starts the TLS connection by telling the website which ciphersuites it supports i.e. it tells the website which types of encryption it is able to use.  The web browser generates a 128-bit random number called a nonce which it sends to the website. The website encrypts this nonce using its RSA private key and sends the encrypted nonce back to the web browser.  The web browser decrypts this encrypted nonce using the trusted-certificate-authority-validated public key belonging to the website. A match confirms that the website knew the private key half of the public key/private key pair and therefore is authentic. |
| 5a) | A digital signature is a cryptographic technique that can be used in a digital world when you want   * to indicate that you are the owner or creator of a document, or * to affirm that you agree with a document’s content or * to verify that the message/document was not tampered with in transit or * to verify that the original document has not been tampered with at source. |
| 5b) | To authenticate that the document did come from the author the receiver uses the author’s public key on the digital signature. The results of this are compared to the contents of the document and if they are the same then the sender knows that the document did come from the author as only the author’s private key could have produced the signature that was successfully decrypted using the author’s public key. |
| 5ci) | A digital signature can be used to tell if a message has been tampered with as if it has then when the author’s public key is used to recover the message from the signature the result will not match the message sent if the message has been tampered with. |
| 5cii) | It can be used to challenge the author as if the message recovered using the author’s public key is the same as the message received then that digitally-signed message could only have been produced by using the author’s private key, this means that they cannot deny that they digitally signed the original message. |
| 6) | With a short message the message itself will be used to create the digital signature, with a longer message a digest (hash) of the message will be used to create the digital signature. |
| 7) | ① Function H is applied to plaintext message to produce message digest  ② Message digest encrypted with X’s private key  ③ Digital Signature + Plaintext Message encrypted with Y’s public key  ④ Encrypted [Digital Signature + Plaintext Message] decrypted with Y’s private key  ⑤ Digital Signature decrypted with X’s public key  ⑥ Message Digest regenerated by applying function H to Plaintext Message  ⑦ Message Digest from Digital Signature compared with regenerated Message Digest  to determine if (a) Plaintext Message has originated with X, and (b) has not been   altered in transit between X and Y |
| 8) | Differences include:   * Viruses are attached to other files whereas Trojans and worms are standalone programs * Viruses and worms can replicate/copy themselves from machine to machine whereas Trojans can’t * Trojans and viruses trick the user into running them, worms do not require any user interaction to run |
| 9a) | Worms can spread by taking advantage of the fact that the same vulnerability appears on many computers as many computers have the same version of the same operating system so this means that they can spread easily. |
| 9b) | Viruses can often be hidden as macro code in word processor/spreadsheet documents. IF macros are enabled then that code can execute when the document is opened. |
| 9c) | If browser software is not up-to-date then fake digital certificates may be used to install a Trojan onto a computer system that is pretending to be another piece of software. |
| 10) | Poor quality code is more likely to have vulnerabilities that can be exploited by malware as it is more likely that the code can be used in a way that it was not intended to be. |
| 11) | Anti-virus and anti-malware software are designed to monitor and protect against attempts to exploit weaknesses in operating systems by hooking deep into the operating system’s core or kernel and function.  Any time the operating system accesses a file, the protection software scans the file to check it is a ‘legitimate’ file or not. If the file is identified as malware by the virus/malware scanner, the access operation will be stopped, the file will be dealt with by the scanner in a pre-defined way and the issue reported to the user. |

9.4 The Transmission Control Protocol/Internet Protocol (TCP/IP)

Chapter 9.4.1 TCP/IP

|  |  |
| --- | --- |
| 1) | Receives data from application layer.  Breaks data down into segments.  Passes segments onto network layer.  Ensure that segments arrive correctly at the other end.  Receive segment from the network layer.  Reassemble received segments in the correct order to form data to pass to application layer. |
| 2) | HTTP  Other possible answers include: HTTPS, FTP, SSH, Telnet, SMTP, POP3, etc… |
| 3) | The original source of the packet and the final destination of the packet are unchanged at each hop, the IP addresses are used to indicate these.  The link layer (MAC) addresses stay the same at each hop. This is because the link layer’s role is to stream bytes between directly connected machines, hosts and routers – and each hop is between two different machines. |
| 4) | The application layer on the host constructs an HTTP GET message. This message is passed to the transport layer which splits it into segments, adding source and destination port numbers. Each segment is passed to the network layer. Source and destination port numbers are added by the network layer and the packet is then passed onto the link layer. The host’s link layer adds the source and destination MAC addresses for the first hop (the MAC address of the host machine and the MAC address for the LAN “side” of the gateway that the host is attached to). The packet then hops from gateway/router to gateway/router until it arrives at the gateway attached for the network that the destination host belongs to.  At each hop the router receives the packet and it is passed from the link layer to the network layer. The network layer inspects the destination IP address and uses this to work out what the next hop should be. The packet is then passed back to the link layer which adds the source and destination MAC addresses (its own and the one belonging to the router/gateway identified as the endpoint for the next hop).  When it arrives at the destination host the link layer will remove the MAC addresses and pass the packet to the network layer. The network layer will remove the IP addresses and pass the segment to the transport layer. The transport layer will combine the segments in the correct order to form the original message and pass this onto the correct application layer process. This process will then deal with the GET request. |
| 5) | A socket is one endpoint of a two-way communication link between two programs running on the network, e.g. a Web browser and a Web server. |
| 6) | A socket is bound to a port number so that the transport layer can identify the application that data is destined for. |
| 7) | Each TCP connection can be uniquely identified by its two endpoints. |
| 8) | On the client side a socket is bound a pair consisting of the IP address of the client (195.61 3.4.7) with a free client port number.  On the server side there will be a listening socket bound to a pair consisting of the IP address of the server (210.56.78.3) and port number 80.  A HTTP GET request will be created on the client side. When received by the server it identifies the TCP/IP connection it belongs to and routes it to corresponding socket. The Web server sends an acknowledgement message back followed by the content of the requested web page. |
| 9) | Well-known port numbers are used by servers to listen for requests for a particular service. Client port numbers are temporary and are allocated to a connection socket requested by a client.  Differences include that client port numbers are only used for the duration of a connection and that well-known port numbers always have the same use e.g. 80 is always used to listen for http requests. |
| 10a) | 192.168.2.22 |
| 10b) | 50268 |
| 10c) | 64.29.1.45.9 |
| 10d) | 80 |
| 10e) | HTTP GET /books.html |
| 11a) | Both lines are http requests for the same image being sent to the same web server using the well-known port number 80. Both requests have been sent from the same IP address but have different port numbers so they are for different processes. |
| 11b) | That after the initial web page has been received there is a need to download an image that is part of the web page so a further GET request has been sent. |
| 12) | A MAC address is a unique physical address for a network adapter. |
| 13) | The MAC addresses of the source and destination network adapters. |

Chapter 9.4.2 Standard application protocols

|  |  |
| --- | --- |
| 1a) | FTP server software |
| 1b) | FTP client software |
| 1c) | The user may need to navigate around the folder structure to find the resource they wish to transfer. Examples of possible commands include: change directory, delete file, delete directory, rename file, rename directory, create directory, etc… |
| 2) | * The user enters a URL into the web browser. * The web browser extracts the FQDN from the URL. * A DNS is used to translate the FQDN into an IP address. * The web browser sends the message GET /books.html to port 80 of the server with the IP address obtained by the DNS server. * The server receives the GET message and accesses the books.html file from its backing store and constructs a response message that includes the content of the html file that is sent back to the client machine. * The web browser receives the response message and displays the web page books.html in a browser window, using the style and structure specified by the HTML. * If the HTML file contains other URLs e.g. for an image, then appropriate GET messages are created and sent to the server. |
| 3) | So that data can be sent in encrypted form between client and server e.g. passwords for log-ins.  So that clients and servers can be authenticated. |
| 4) | Port 443 is used instead of port 80.  There is an SSL sublayer which encrypts data passing from the application layer to the transport layer / decrypts data passing from the transport layer to the application layer. |
| 5) | Two possible answers:  DELE no to delete the email specified by no  RETR no to retrieve the email specified by no (so that the user can read the message) |
| 6) | Because it provides additional security e.g. encryption of transmitted data. |

Chapter 9.4.3 IP address structure

|  |  |
| --- | --- |
| 1a) | 204.63.85.63 |
| 1b) | 217.135.25.19 |
| 2a) | 16 |
| 2b) | 23 |
| 2c) | 19 |
| 2d) | 25 |
| 3) | b |
| 4) | The router receives the datagram and searches its routing table to determine how to route the datagram within the University of Kent. It will do this by looking at the first 23 bits of the destination IP address to determine which subnet to forward the datagram onto. |

Chapter 9.4.4 Subnet masking

|  |  |
| --- | --- |
| 1) | 192.168.2.0  192.168.0.0  192.168.252.0 |
| 2) | By applying the subnet mask to its own IP address and to the IP address of the destination it will know that the destination is not on the same subnet as itself and so the datagram is sent to the default gateway. |

Chapter 9.4.5 IP standards

|  |  |
| --- | --- |
| 1a) | 232 |
| 1b) | 2128 |
| 2) | Because the number of IPv4 addresses is too limited and they were running out. |

Chapter 9.4.6 Public and private IP addresses

|  |  |
| --- | --- |
| 1a) | Where the hosts will be communicating only with other hosts in the same network e.g. internal communications within a large bank.  Arrival/departure boards at an airport do not need to be accessed from other networks.  Where security is important, accessed to the Internet is handled by application layer gateways so hosts in an organization do not need direct access to the Internet. |
| 1b) | It allows an organization to use far more address space than would be possible if they had to use routable addresses. |
| 2) | 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16 |
| 3) | Non-routable addresses are private and are not globally unique – they can be reused on any TCP/IP network. Datagrams that contain non-routable source or destination IP addresses will be rejected by routers. Routable addresses are globally unique. |

Chapter 9.4.7 Dynamic Host Configuration Protocol (DHCP)

|  |  |
| --- | --- |
| 1) | The primary purpose of DHCP is to automate the setting up of hosts that are connecting to a TCP/IP network. |
| 2) | ISP providing access to the Internet for a large number of residential customers, only some of whom will be online at any particular time.  Wireless LAN. |
| 3) | To allocate IP addresses to hosts via automatic allocation of a permanent IP address, dynamic allocation of a temporary IP address and conveying a manually assigned IP address to a host.  To deliver host-specific configuration parameters such as subnet mask to a host. |
| 4) | A DHCP server is needed to maintain a list of available IP addresses. It “listens” for DHCP discover messages and when one is received it sends out a DHCP offer message which contains an available IP address that the client machine can use. |

Chapter 9.4.8 Network Address Translation

|  |  |
| --- | --- |
| 1) | Because IP datagrams forwarded beyond the home LAN into the global Internet cannot use non-routable addresses because these addresses will be rejected by Internet routers. |
| 2) | The NAT-enabled router maintains a table of port numbers and the associated LAN host IP address and port number. This means that when it receives a datagram it can look at the destination port number of the datagram and forward it on to the relevant LAN-host.  When sending datagrams to a device outside the LAN the router uses its own routable IP address as the source address with a port-number.  This means that all communications between the LAN and the WAN are actually between the WAN and the router. |
| 3) | In TCP/IP the port number is used to indicate which process the datagram is for; in NAT protocol the port number is used to indicate which host and which process. |
| 4a) | Because the connection is not established by the two end-points, the NAT-enabled router establishes the connections instead of the client (the end-point). |
| 4b) | As there are so many more IP addresses there are enough routable IP addresses available for all devices that wish to connect to the Internet – therefore they may not have to use non-routable IP addresses. |

Chapter 9.4.9 Port forwarding

|  |  |
| --- | --- |
| 1) | When clients in other LANs connected to the Internet want to reach servers assigned private unregistered IP addresses and located behind a NAT router in another LAN. |
| 2) | There will be an entry in the port mapping table with a LAN side entry of 172.31.78.4:80 and a WAN side entry of 146.31.18.97:80. |
| 3) | Because dynamic IP addresses were being used and a different IP address has now been allocated to the Web server. To solve this problem by getting DHCP to assign static IP addresses to the servers. |

Chapter 9.4.10 Client server model

|  |  |
| --- | --- |
| 1) | The client-server model is the most commonly employed of the architectural styles for when one application interacts with another. A server is one application, a client is another. A server application listens for requests, from client applications, for services which it offers and the client applications consume these services. |
| 2) | REST stands for Representational State Transfer.  Central to the concept of REST is the notion of resources. Resources are represented by URIs.  Another key aspect of the design of REST web services is that resources should be linked together and representations of these resources should enable a user to move from one resource to another by following these links.  Amazon’s website is an example of a web service based on REST. |
| 3) | a) HTTP is an example of a uniform interface, the method of communication between clients and the server is always the same.  b) It is not the resource itself that is sent to the client, but a representation of the resource. Two ways of representing a resource are JSON and XML.  c) Every resource has an ID; a URI is one way that an ID for a resource can be represented.  d) Connections between resources are possible e.g. “Likes” on Facebook are a way of connecting different nodes in Facebook’s social network graph. |
| 4) | Creating, retrieving, updating, deleting – the four operations that can be done with a web resource. |
| 5) | GET 🡪 SELECT  POST 🡪 INSERT  DELETE 🡪 DELETE  PUT 🡪 UPDATE |
| 6) | JSON is easier for humans to read.  JSON is quicker for machines to parse.  JSON is easier for a machine to generate. |
| 7) | XML is text-parsing, not code-executing, which means that it won’t accidently run malicious code.  XML allows namespaces meaning it can support multiple instances of the same field name. |
| 8) | Because it allows communications where the server sends data when it is available i.e. real-time event-driven applications; HTTP only allows for client-initiated communications. |

Chapter 9.4.11 Thin- versus thick-client computing

|  |  |
| --- | --- |
| 1) | Approx. 5250 MiB will be required for the server supporting 100 thin clients. |
| 2a) | Reduced power consumption will save money.  Lower hardware costs.  Longer life-expectancy for thin-client machines means that replacement costs will occur less frequently.  Potential to save money with cheaper software licencing. |
| 2bi) | They may need to use high-end graphics processing which will run better on thick clients. |
| 2bii) | They may wish to develop a wide range of different applications and some of these may be processor intensive. |
| 3) | Because the machines won’t need to be reset individually, changes only need to be to the server rather than to each individual machine. |
| 4) | Because they provide input and receive results from application software but don’t do any significant processing themselves, |

10 Fundamentals of databases

Chapter 10.1 Conceptual data models and entity relationship modelling

|  |  |
| --- | --- |
| 1a | Examples include: Ward, Patient, Doctor |
| 1b | Examples include: Member, Loan, Item |
| 1c | Examples include: Athlete, Event, Result |
| 1d | Examples include: Item, Order, Customer |
| 2a | Examples include: Surname, PatientID |
| 2b | Examples include: WardName, NumberOfBeds |
| 2c | Examples include: Surname, Forename |
| 3 | D. Exactly one |
| 4 | (d) Registration number because it is unique for every car |
| 5 | (a) is not suitable because two guests could have the same surname  (b) is not suitable because the same person could make more than one booking  (d) is not suitable as Surname of Guest is not needed – so it is not minimal |
| 6a | PatientID or NHSNumber |
| 6b | WardName or WardNumber |
| 6c | StaffNumber |
| 7 | Ward (WardName, NumberOfBeds)  Nurses(StaffNumber, Surname, Forename) |
| 8 | Aeroplane (AeroplaneNo, Type, NumberOfSeats)  FlightStaff (StaffNumber, Name, Specialism) |
| 9a | ii |
| 9b | iii |
| 9c | iv |
| 9d | i |
| 10ai | A property of an entity |
| 10aii | A link between two entities |
| 10b | Ward – WardName  Nurse – StaffNumber  Patient – PatientID  Consultant – StaffNumber |
| 10c | Ward  Nurse  Consultant  Patient  Patient  Ward |

Chapter 10.2 Relational databases

|  |  |
| --- | --- |
| 1 | {Sarah, CS}  {Sarah, Physics}  {Sarah, Maths}  {Jim, CS}  {Jim, Physics}  {Jim, Maths}  {Kevin, CS}  {Kevin, Physics}  {Kevin, Maths} |
| 2 | (d) GalaNo, RaceNo because it is unique and minimal |
| 3a | (i) CompactDiscId, SongId could have same song on CD more than once, different singer therefore not unique in collection  (ii) TrackNo, SongId song could be on same track on different CDs(different CompactDiscIds) therefore not unique in collection  (iii) SongIc could be on many CDs therefore not unique in collection |
| 3b | Composite primary key |
| 4 | Ward(WardName, NoOfBeds)  Nurse(StaffNo, Name, *WardName*) |
| 5 | GP(GPId, GP)  Patient(PatientNo, Name, HomeAddress, DateOfBirth, Gender, NHSNo, *GPId*) |
| 6a | Team  EventTeam  Event |
| 6b | Event(EventId, EventDescription, Date, Time)  Team(TeamId, TeamName, ContactTelNo)  EventTeam(EventId, TeamId) |
| 7a | NursePatient  Patient  Nurse |
| 7b | Ward(WardName, NoOfBeds)  Patient**(**PatientNo, Name, Address, DateOfBirth, Gender, *WardName*, *StaffNo*)  Nurse(StaffNo, Name, Rank, *WardName*)  NursePatient(StaffNo, PatientNo)  Consultant**(**StaffNo, Name, Specialism) |
| 8a | If cars change drivers frequently then not just *DriverId* will need updating in **Car** relation but also all the other details of drivers. |
| 8b | If there is little use made, let’s say, of the foreign key *DriverId* in the **Car** relation, i.e. when looking up car information, driver information is not required then *DriverId* could be dropped from **Car** relation. Foreign key *CarId* would only be needed in **Driver** relation |
| 9 | **Teacher** (TeacherId, Name, SubjectId, ManagerId) |
| 10a | BloodDonor(DonorId, Surname, Forename, Address, TelNo, DOB, BloodType)  BloodGiven(VesselId, Date, Quantity, *DonorId*) |
| 10b | BloodDonor  BloodGiven |
| 10ci | A collection of tables. |
| 10cii | An attribute in one table that is a primary key in another table. |
| 10ciii | By the foreign key *DonorId*. |
| 10civ | Same as entity descriptors in 10a |

Chapter 10.3 Database design and normalisation techniques

|  |  |
| --- | --- |
| 1 | GpId → GpName PatientId → PatientName PatientId → GpId  PatientId → GpName  **GP** (GpId, GpName)  **Patient** (PatientId, PatientName, GPId) - remove determinant *PatientId* to its own relation because it cannot be the primary key in **GP** relation because it is not unique. Remove what depends upon *PatientId* to **Patient** unless it also depends upon something else as well. So remove *PatientName* but not *GPName*. |
| 2a | It contains a repeating group (Patient) |
| 2b | **Patient** (PatientID, WardName, PatientName)  **Ward** (WardName, WardType) |
| 3a | StudentId |
| 3b | TeacherId |
| 4a | It contains a repeating group (CustomerID). |
| 4b | **Salesman** (SalesmanID, SalesmanName)  **SalesmanCustomer** (SalesmanID, CustomerID) |
| 5 | **GP** (GPID, GPName)  **GPPatient** (GPID, PatientID) |
| 6a | Yes, TutorName, e.g Ainsley in row 2 can be inferred from row 1. |
| 6b | No, primary key is *TutorNo, StudentNo* but *TutorName* is a fact about *TutorNo* not *TutorNo, StudentNo*. Similarly, *StudentName* is a fact about *StudentNo* not *TutorNo, StudentNo*. |
| 7a | The WardType for each Ward is stored more than once. |
| 7b | **Patient** (PatientNo, PatientName, WardName)  **Ward** (WardName, WardType) |
| 7c | *PatientNo* is the primary key for the **Patient** relation, *WardName* is the primary key for the **Ward** relation. |
| 7d | Yes – there are no repeating groups, no transitive dependencies and no partial dependencies. Or, every non-key attribute is a fact about the key the whole key and nothing else. |
| 8a | **GP**(GPId, GPName, PatientId, PatientName). Removed repeating group. |
| 8b | **GP** (GPId, GPName)  **Patient** (PatientId, PatientName, GPId).  GPName depends upon GPId not GPId, PatientId, the key in the 1NF **GP** relation. PatientName depends upon PatientId not GPId, PatientId. So eliminate partial dependencies by splitting the original relation into two separate relations as shown. |
| 8c | **GP** (GPId, GPName)  **Patient** (PatientId, PatientName, GPId).  There are no dependencies between non-key attributes so the 2NF relations are already in 3NF. |
| 9a | **TeacherTeaches** (TeacherId, TeacherName, StudentId, StudentName, SubjectName) |
| 9b | **TeacherTeaches** (TeacherId, StudentId, SubjectName)  **Student** (StudentId, StudentName)  **Teacher** (TeacherId, TeacherName) |
| 9c | **Same as 9b** |
| 10 | **Swimmer** (SwimmerNo, Name)  **Gala** (GalaNo, Venue)  **Stroke** (StrokeNo, StrokeName)  **GalaRace** (GalaNo, RaceNo, StrokeNo)  **RaceEntry** (GalaNo, RaceNo, SwimmerNo) |
| 11 | **Manager** (ManagerId, ManagerName)  **Band** (BandId, BandName)  **Booking** (BandId, BookingDate, VenueId)  **Venue** (VenueId, VenueName) |
| 12a | (i) PatientId |
| 12b | PatientId → PatientName  PatientId → GpId  PatientId → GpName  GpId → GpName |
| 12c | Every determinant should be a candidate key, but *GPId* is a determinant but not a candidate key in relation **Patient** so make it a candidate key by placing it in its own relation **GP** and removing its functionally dependent attribute *GPName* from **Patient**.  **Patient** (PatientId, PatientName, GPId)  **Gp** (GpId, GpName) |
| 12d | 3NF and BCNF are equivalent for a relation with only one candidate key.  Relations **Patient** and **GP** each have only one candidate key. |
| 13a | SwimmerNo → Name  GalaNo → Venue  StrokeNo → StrokeName  GalaNo, RaceNo → StrokeNo  GalaNo, RaceNo → StrokeName  GalaNo, RaceNo, SwimmerNo → Time |
| 13b | SwimmerNo  GalaNo  StrokeNo  GalaNo, RaceNo  GalaNo, RaceNo, SwimmerNo |
| 13c | SwimmerNo |
| 13d | **Swimmer** (SwimmerNo, Name)  **Gala** (GalaNo, Venue)  **GalaRace** (GalaNo, RaceNo, StrokeNo)  **Stroke** (StrokeNo, StrokeName)  **RaceEntry** (GalaNo, RaceNo, SwimmerNo, Time) |

Chapter 10.4 Structured Query Language (SQL)

|  |  |
| --- | --- |
| 1 | SELECT StudentName  FROM Student  WHERE Gender = 'F'; |
| 2 | SELECT Ward.NurseInCharge, Patient.Surname, Ward.WardName  FROM Patient, Ward  WHERE Ward.WardName = Patient.WardName; |
| 3 | SELECT Ward.NurseInCharge, Patient.Surname  FROM Patient, Ward  WHERE Patient.WardName = Ward.WardName  ORDER BY Surname DESC; |
| 4 | SELECT Ward.WardName, Patient.Surname  FROM Ward, Patient  WHERE Ward.WardName = Patient.WardName  AND PatientId > 1  ORDER BY Surname ASC; |
| 5 | SELECT WardName, Surname  FROM Patient  WHERE PatientId <= 3  ORDER BY PatientId DESC; |
| 6 | Ecuador, 10600000, 455502  El Salvador, 5300000, 20865  Guyana, 800000, 214969 |
| 7 | Santiago, 13200000  Havana, 10600000  Georgetown, 800000 |
| 8 | SELECT Capital, Population  FROM Country  WHERE Name = 'Chile' OR Name = 'Cuba' OR Name = 'Guyana'; |
| 9 | DELETE FROM Borrower  WHERE BorrowerId = 3; |
| 10 | DELETE FROM Country  WHERE Population > 15000000; |
| 11 | INSERT INTO Ward  VALUES ('Amersham', 'Sister Brody', 25); |
| 12 | INSERT INTO Country (Name, Capital)  VALUES ('UK', 'London'); |
| 13 | UPDATE Country  SET Population = 64100000,  Area = 243610  WHERE Name = 'UK'; |
| 14 | CREATE TABLE Borrower  (  BorrowerId SMALLINT PRIMARY KEY,  Surname VARCHAR (25),  Initial VARCHAR (6)  );  CREATE TABLE BooksOnLoan  (  ISBN INTEGER (13),  CopyNo SMALLINT,  BorrowerId SMALLINT,  DateDueBack DATE,  CONSTRAINT ISBNCopyNoPK PRIMARY KEY (ISBN, CopyNo),  FOREIGN KEY (BorrowerId) REFERENCES Borrower(BorrowerId)  ); |

Chapter 10.5 Client server databases

|  |  |
| --- | --- |
| 1) | A multi-user client server database system where data items are stored in a database on a server which may be accessed simultaneously by programs running on client workstations. |
| 2) | Means access occurs at the same time or over the same time interval. |
| 3) | An update is lost when two transactions that access the same database item have their operations interleaved in such a way that makes the value of the database item incorrect or inconsistent. |
| 4) | A record lock is a concurrency control method which can prevent a lost update. The lock ensures exclusive-access to a record when it is being updated. Other transactions are blocked from accessing the record until after it has been updated and the change permanently recorded. The blocked transactions then see the updated record.  Serialisation attempts to serialise access to a data item in order to detect and prevent the lost update problem occurring. Transactions attempting to alter a data item that is currently the subject of another transaction are detected and aborted. Any temporary changes cancelled.  A timestamp is a unique identifier created by a database server that indicates the relative starting time of a transaction. The database records the transaction timestamp of the last transaction to read data item X and the transaction timestamp of the last transaction to write data item X. The database server applies rules using these to determine if a transaction’s actions will result in the integrity of the database being compromised. If it will the server aborts the transaction. |
| 5) | Commitment ordering can be used in a distributed database system such as mobile device client server systems where transactions are created locally on the mobile device before being sent to the server for execution. The transactions are scheduled at the server so that they are committed in an order that avoids concurrency conflict. |

11 Big Data

Chapter 11.1 Big Data

|  |  |
| --- | --- |
| 1) | Big data is data that can’t be processed or analyse using traditional processes or tools because it is too big to fit into a single server, too heterogeneous or its production occurs at a very high rate. |
| 2) | Volume  Velocity  Variety |
| 3) | Volume refers to the size of the data to be processed. Big data often has volume of hundreds of terabytes or petabytes that need to be analysed as a single dataset. |
| 4) | Data is distributed over more than one server and processing of the data is done in parallel (executed on more than one machine at the same time). |
| 5) | Blocks of individual files are spread across more than one server. |
| 6) | Fault-tolerant means that if one component of the system fails then the system can continue to function correctly despite the failure of that component e.g. each block of data is stored on several servers so if a server fails then the data can still be retrieved from one of the alternative servers. |
| 7) | In HDFS each block of data is written three times with at least one copy of the block being written to a different server. |
| 8) | Scalable means that more servers can be added as the size of the file grows – the racks of servers are all interconnected and there is a file system that keeps track of which blocks belong to which file and where the blocks are stored. |
| 9) | The data containing the text to count words in is stored on a distributed file system.  A copy of the Map function is sent to each server storing blocks of the distributed file.  The Map function would produce a list of keyword pairs for each word it encounters in the block of text.  The key-word pairs would be shuffled and sorted to group them together.  A reduce function would be used to collate the keyword pairs and produce a total count for each word. |
| 10) | Variety means that the data can be structured, semi-structured or unstructured. Big data tends to be either semi-structured or unstructured. |
| 11) | Velocity refers to whether the data is in motion and its frequency or if the data is at rest. Big Data often has data arriving at a very high rate from multiple sources simultaneously. Big Data can be “at rest” which means it is a large data set which tends to be processed by batch processing; data in motion is data that is streamed at some frequency continuously (i.e. large amounts of new data is constantly arriving). |
| 12) | Machine learning techniques can be used with Big Data to discern patterns in data and to extract useful information from large datasets. One method of doing this is to have a predictive model which can then be used in the algorithm that processes streaming data to extract relevant values from the data in the stream.  Examples:   1. Data about Google searches being made were used to predict the spread of a winter flu virus. 2. Predictive models are also used in fraud detection on credit card transactions. |
| 13a) | State cannot be modified after it is created i.e. after a data structure has been given a value its value cannot be changed. |
| 13b) | Statelessness refers to functions that are pure – they can be reasoned about mathematically and always return the same value for a given input. There is no current state that can also be altered by the function. |
| 13c) | Higher-order functions are functions that take other functions as arguments. |
| 14) | Immutable data structures  Statelessness  Higher-order functions |
| 15) | The fact-based model is a way of modelling data so that each fact captures a single-piece of information. The data is stored as atomic facts, these facts are kept immutable as a timestamp is associated with each fact (indicating that this fact was true at the indicated time, even though the fact may not have been true at other times). Each fact is identifiable using a unique randomly-generated number so that query processing can identify duplicates. |
| 16) | Atomic  Time-stamped |
| 17) | An atomic fact is one that can’t be further subdivided into anything meaningful. |
| 18) | Each fact represents either a piece of information about an entity or a relationship between entities. |
| 19a) | Nodes represent entities in the system. |
| 19b) | Edges represent relationships between nodes/entities. |
| 19c) | Properties are information about entities. |
| 20) |  |
| 21) | Timestamps would be used to indicate when the messages were posted on the bulletin board and would also be used to differentiate between multiple messages posted by the same person. |
| 22) | Fact-based models are continuously growing so Big Data techniques are needed to cope with the growing list of facts – more than one server is typically going to be required. |
| 23) | Three advantages are:  It is very easy to add new types of information to the schema.  Data is true for ever.  Queries can be made about historical data because facts are immutable (data is never deleted). |

12 Fundamentals of functional programming

12.1 Functional programming paradigm

Chapter 12.1.1 Function type

|  |  |
| --- | --- |
| 1) | Cube the input. |
| 2) | Double the input. |
| 3)a) | Function as process: make a printed copy of the sheet  Function as object: the machine (photocopier) |
| 3)b) | Function as process: peel the potatoes  Function as object: the kitchen tool (potato peeler) |
|  | **Practical activities**  double :: Integer -> Integer  double x = 2\*x  cube :: Integer -> Integer  cube x = x\*x\*x  timesten :: Integer -> Integer  timesten y = y\*10 |

Chapter 12.1.2 First-class object

|  |  |
| --- | --- |
| 1 | Objects which can  • appear in expressions  • be assigned to a variable  • be assigned as arguments to functions  • be returned by function calls |
| 2a | 1024 |
| 2b | 59049 |
| 2c | 9765625 |
|  | Programming task   1. Should display 55 2. Should display 225 3. Should display 15   The first call produces a value of 55 by a recursive call. To work out summy(square) 5 it adds the result of summy(square) 4 to the result of square(5). In order to work out the result of summy(square) 4 it adds the result of summy(square) 3 to square(4), etc… Eventually it will reach the base case of summy(square) 0 which returns a value of 0 and does not recurse.  The end result of this function call is to sum the squares of all the positive integers from 5 downwards (25+16+9+4+1).  summy(cube) 5 is exactly the same except instead of using the square function it uses the cube function and so is adding up the cubes of all the positive integers from 5 downwards (125+64+27+8+1).  summy(identity) 5 is the same except instead of using the square or cube functions it uses the identity function and so is adding up all the positive integers from 5 downwards (5+4+3+2+1). |

Chapter 12.1.3 Function application

|  |  |
| --- | --- |
| 1a | 9 |
| 1b | 25 |
| 2 | n |
| 3 | 3 and 5 |
| 4 | When a function is applied to a particular argument(s). |

Chapter 12.1.4 Partial function application

|  |  |
| --- | --- |
| 1)  *computerProg1*  Input  Output |  |
| 2) | *multiply10*  y |
| 3) | It means that the function takes an integer and returns a function. The function returned takes an integer and returns an integer. |
| 4) | Because a radio takes two parameters (frequency and radio waves), when you have selected a frequency you then have a more specialised function (a radio for just the one frequency) that takes just one parameter (radio waves). |
| 5) | As the first one takes two integers and returns an integer but the second takes an integer and returns a function (that takes an integer and returns an integer). The second one is a partial function application of the first function to just one of its parameters. |
|  | Programming Task 1  multiplyxy :: Integer -> Integer -> Integer  multiplyxy x y = x \* y  multiply2y :: Integer -> Integer  multiply2y = 2 \* y |
| 6) | Number 1 |
| 7) | Number 1 |

Chapter 12.1.5 Composition of function

|  |  |
| --- | --- |
|  | Programming Task 1   1. (double.square) 6 2. (square.double) 6 3. double (square 6) for a); square (double 6) for b |
|  | Programming Task 2  add (square 2) (square 3) |
|  | Programming Task 3  sqrt (add (square 4) (square 3)) |
| 1) | Functional composition combines two functions to get a new function. |
| 2) | (x-3)3 |
| 3) | Because the co-domain of function f is not the same as the domain of function g. |
| 4) | 91 |

12.2 Writing functional programs

Chapter 12.2.1 Functional language programs

|  |  |
| --- | --- |
|  | Programming Task 6  a)  (\x -> x \* 2) 5  b)  (\x -> x \* 2 + 3) 5 |
|  | Programming Task 7  (\x y -> if x == 2 \* y then True else False) |
|  | Programming Task 8  a)  let dbl x = 2 \* x  b)  let cube x = x^3 |
|  | Programming Task 9  cube :: Integer -> Integer  cube x = x\*x\*x  double :: Integer -> Integer  double x = 2\*x |
| 1)a) | [1,4,9,16] |
| 1)b) | ["SLAP!","BAM!","WALLOP!"] |
| 2) | [5,6] |
| 3) | 180 |
| 4) | 78 |

12.3 Lists in functional programs

Chapter 12.3.1 List processing

|  |  |
| --- | --- |
| 1a | 9 |
| 1b | 25 |
| 2 | n |
| 3 | 3 and 5 |
| 4 | When a function is applied to a particular argument(s). |