# Comp1 Exam Preparation Pack Secret Messages Program

### Aim:

The aim of the document is to help YOU understand the preliminary code and prepare for the COMP1 Exam on 3rd June 2013. This pack specifically prepares you for sections C and D of the Exam

**Section C**

You are advised to spend no more than **20 minutes** on this section. Questions will refer to the **Preliminary Material** and the **Skeleton Program**, but will not require programming.

**Section D**

You are advised to spend no more than **50 minutes** on this section. Questions will use the **Skeleton Program** and the **Preliminary Material**

### Instructions

You need to complete this document on the computer and add rows to the tables below. Feel free to make the document landscape if it helps. Make sure you have saved a copy into your user area BUT beware… I may add things to the tasks below as stuff gets spotted with the code.

1. Complete the table listing all Global variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable identifier** | **Data type** | **Line numbers where Used** | **Sample data** |
| Ciphertext | String | 146 | Khoor Zruog |
| Plaintext | String | 147 | Hello World |
| Choice | String | 148 | a |
| StartPosition | Integer | 149 | 1 |
| EndPosition | Integer | 150 | 1000 |
| AmountToShift | Integer | 164 | 3 |
| SizeOfRailFence | Integer | 169 | 2 |
| N | Integer | 184 | 100 |

1. Complete the table listing all Local variables in the main program,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable identifier** | **Data type** | **Sub routine Identifier** | **Line numbers where used** | **Sample data** |
| TextFromUser | String | GetTextFromUser | 31 | Hello |
| CurrentFile | File | GetTextFromFile | 40 | file |
| ByteFromFile | Byte | GetTextFromFile | 42 | b'n' |
| TextFromFile | String | GetTextFromFile | 43 | abc |
| Key | Integer | GetKeyForCaesarCipher | 51 | 3 |
| ASCIICode | Integer | GetTypeOfCharacter | 55 | 4 |
| TypeOfCharacter | String | GetTypeOfCharacter | 56 | Upper |
| NewASCIICode | Integer | ApplyShiftToASCIICodeForCharacter | 67 | 5 |
| ChangedText | String | UseCaesarCipher | 75 | abc |

1. Complete the table listing all Procedures and functions

|  |  |  |  |
| --- | --- | --- | --- |
| **Procedure / function Name and line number** | **Brief description** | **Value parameters** | **Reference parameters** |
| DisplayMenu (L6) | Print out the menu items available | None | None |
| GetMenuChoice (L25) | Gets user input of their menu choice | None | None |
| GetTextFromUser (L30) | Gets text input from user and return | None | None |
| GetPositionsToUse (L34) | Gets user input for start and end point to use in a file | None | None |
| GetTextFromFile (L9) | Opens a file and returns the content between defined start and end positions | StartPosition, EndPosition | None |
| GetKeyForCaesarCipher (L50) | Gets user input for the number of shifts to use for a Caesar cipher | None | None |
| GetTypeOfCharacter (L54) | Returns whether the passed character is an uppercase letter, lowercase letter or other character | ASCIICode | None |
| ApplyShiftToASCIICodeForCharacter (L63) | Shifts an ascii character code by a defined amount | ASCIICode, AmountToShift | None |
| UseCaesarCipher (L74) | Perform Caesar cipher on a given piece of text, based on given shift | OriginalText, AmountToShift | None |
| GetSizeOfRailFence (L82) | Gets user input for number of lines to use for a rail fence | None | None |
| EncryptUsingRailFence (L86) | Encrypt given text with railfence based on given number of lines | OriginalText, SizeOfRailFence | None |
| DecryptUsingRailFence (L95) | Decrypt railfence ciphertext based on size | Ciphertext, SizeOfRailFence | None |
| GetValueForN (L123) | Gets user input for n value | None | None |
| EveryNthCharacterSteganography (L127) | Retrieves hidden message using steganography | StartPosition, EndPosition, N | None |
| DisplayPlaintext (L135) | Prints out plaintext currently stored in the global variable | TextToDisplay | None |
| DisplayCiphertext (L140) | Prints out ciphertext currently stored in the global variable | TextToDisplay | None |

1. Give examples of a variable identifier for each of the following “Variable Roles” .. The \* shows that the role is given as an example in the textbook. This website also has good examples: <http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html>   
   More info here:  
   <https://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Programming/The_Role_of_Variables>

|  |  |
| --- | --- |
| Role (see page 66 of book or follow links) | Line numbers of examples… Copy and paste code Highlighting the variable being used in that role. |
| [fixed value](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html" \l "fixed%20value) \* | **NoOfCiphertextCharacters** (L97)  def DecryptUsingRailFence(Ciphertext, SizeOfRailFence):  Plaintext = ''  NoOfCiphertextCharacters = len(Ciphertext)# created/assigned  ……. # used  return Plaintext # destroyed |
| [stepper](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#stepper) \* | **Count** (L41)  for Count in range(1, StartPosition): |
| [follower](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#follower) \* | **Count2** (L89)  Count2 = Count1 |
| [most-recent holder](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#most-recent) \* | **ASCIICode** (L77)  for Count in range(0, len(OriginalText)):  ASCIICode = ord(OriginalText[Count]) |
| [most-wanted holder](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#most-wanted) \* | **AmountToReduceNoOfColumnsTimesjBy** += 1 (L114) |
| [gatherer](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#gatherer) \* | **Ciphertext** (L91)  Ciphertext += OriginalText[Count2] |
| [one-way flag](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#one-way) | None |
| [temporary](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#temporary) \* | **ByteFromFile** (L45)  ByteFromFile = CurrentFile.read(1)  TextFromFile += ByteFromFile.decode('ASCII') |
| [organizer](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#organizer) | No examples of this role  (Plaintext and Ciphertext in function DecryptUsingRailFence act a little like an organizer) |
| [container](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#container) | No examples of this.  (Plainttext and Ciphertext are string datatypes but can also be thought of as data structures to which we add values/characters). |
| [walker](http://cs.joensuu.fi/pages/saja/var_roles/stud_vers/stud_Python_eng.html#walker) | No examples |

1. Complete the table to list all of the selection statements in the code

|  |  |  |  |
| --- | --- | --- | --- |
| **Line Number start** | **Line Number End** | **Type (IF/ ELIF / ELSE)** | **Criteria**  **(treat elif as a separate line in this table)** |
| 55 | 56 | IF | ASCIICode >= ord('A') and ASCIICode <= ord('Z') |
| 57 | 58 | ELIF | ASCIICode >= ord('a') and ASCIICode <= ord('z') |
| 59 | 60 | ELSE | N/A |
| 65 | 69 | IF | TypeOfCharacter != 'Other' |
| 66 | 67 | IF | TypeOfCharacter == 'Upper' |
| 68 | 69 | ELSE | N/A |
| 70 | 71 | ELSE | N/A |
| 112 | 114 | IF | LastFullRowNo != 0 |
| 113 | 114 | IF | j > LastFullRowNo |
| 119 | 120 | IF | NoOfCiphertextCharactersProcessed <= NoOfCiphertextCharacters |
| 154 | 155 | IF | Choice == 'a' |
| 156 | 157 | ELIF | Choice == 'b' |
| 158 | 159 | ELIF | Choice == 'd' |
| 160 | 161 | ELIF | Choice == 'e' |
| 162 | 166 | ELIF | Choice == 'g' |
| 167 | 171 | ELIF | Choice == 'h' |
| 172 | 176 | ELIF | Choice == 'j' |
| 177 | 181 | ELIF | Choice == 'k' |
| 182 | 186 | ELIF | Choice == 'n' |
| 187 | 188 | IF | Choice != 'q' |

1. Complete the table to list all of the iteration (looping) statements in the code

|  |  |  |  |
| --- | --- | --- | --- |
| **Line Number start** | **Line Number End** | **Type  (For,  Do while,  Do until)** | **Criteria for exit of loop.**  **Fixed🡪 state the number of loops or the stepper variable name**  **Logic 🡪 state the Boolean expression to exit** |
| 41 | 42 | For | Fixed: Stepper Count in range(1, StartPosition) |
| 44 | 46 | For | Fixed: Stepper Count in range(StartPosition, EndPosition + 1) |
| 76 | 79 | For | Fixed: Stepper Count in range(0, len(OriginalText)) |
| 88 | 92 | For | Fixed: Stepper Count1 in range(0, SizeOfRailFence) |
| 90 | 92 | While | Fixed: Bool: Count2 < len(OriginalText) |
| 109 | 120 | For | Fixed: Stepper i in range (0, NoOfColumns) |
| 111 | 120 | For | Fixed: Stepper j in range (0, NoOfRows) |
| 130 | 132 | While | Fixed: CurrentPosition <= EndPosition |
| 151 | 188 | While | Logic: Choice != 'q' |

1. List all of the BUILT-IN functions used in the code (not that many!!!)

|  |  |  |
| --- | --- | --- |
| **Line Number** | **Function identifier** | **Description of usage (include any parameter input and the expected output** |
| 7 | Print | Outputs text to shell/console. Single parameter representing this text. |
| 31 | input | Gets text input from the user. Single parameter representing text to display as the input prompt. |
| 40 | open | Opens a file. First parameter is filepath, second is mode to open in (read, write etc.) |
| 42 | read | Returns byte from a file at specific position (indicated by single integer parameter). |
| 46 | decode | Decodes a string based on specified encoding method. Returns the result. |
| 47 | close | Closes a file to free up memory |
| 55 | ord | Returns the Unicode integer representation from the single character parameter |
| 76 | range | Creates an array of ascending integers between a start integer (parameter 1) and 1 less than end integer (parameter 2) |
| 76 | Len | Returns the length of a string passed as the single parameter |
| 79 | chr | Returns the string of a Unicode code integer, which is the single parameter |

1. Complete the table to list all of the places a user has to INPUT a value: Include a description of valid data with examples and invalid data with the name of the error caused.

|  |  |  |  |
| --- | --- | --- | --- |
| **Line Number** | **User Input Description** | **Valid data description with examples** | **Invalid data description with the name of the error that will be caused** |
| 26 | User must enter a letter representing a menu choice | Any letter defined on the menu  (a,b,d) | Z  Will not throw an error, but no menu item will be selected |
| 31 | Accepts text from a user  Range check is done of ascii (lines 55-60) | String  (hello world) | None |
| 35 | Asks user for an integer start position in file to find hidden message using stenography | Any integer from 1 to number of characters in the file  (1,2,3) | A string eg. “x” will cause a runtime error, as it cannot be converted to an integer. Need to do a type check. |
| 36 | Asks user for an integer end position in file to find hidden message using stenography | Any integer from start position to number of characters in the file  (100, 200, 1000) | ^ |
| 51 | Request shift key value to use in the Caesar cipher | Between -25 and -1, and between 1 and 25 | ^ |
| 83 | Test number of lines in rail fence encryption | A number between 1 and length of the plain text | ^ |
| 124 | Test character position ‘jump’ value (N) in file for Stenography | A number between 1 and difference between start and end position (10, 20) | ^ |
| 188 | Hit enter to choose next menu option | Enter key | None |

1. Complete the table to list all of the places that the code is performing a validation test. You will have already listed these lines in either the selection or loop statements)

|  |  |  |
| --- | --- | --- |
| **Line Number** | **Validation Type (see pages 96🡪98 in AS book)** | **Description of Validation inc. criteria (paste code)** |
|  |  | **NONE** |
|  |  |  |

1. Complete the table to produce a simple input test plan

***Erroneous*** – is correct type, but out of range, not on the list etc..  
**Invalid** – wrong type of data being entered.   
**Normal** – Data the program would expect.   
**Boundary – Test the system around the start and end of the data range.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INPUT Line Number** | **Description of test** | **Normal Data** | **Erroneous Data** | **Boundary Data** | **Invalid Data** |
| 26 | Test menu option choice | ‘a’, ‘b’, ‘d’, ‘e’, ‘g’, ‘h’, j’, ‘k’, ‘n’ | ‘z’ | ‘a’ and ‘n’. Not really appropriate as is not a strict range | Number, date, boolean |
| 31 | Get text (plain text and cipher text) | Alphanumeric text and punctuation. | Non-printing characters | A, z, 0, 9 | None |
| 35 | Test entry of start position in file for Stenography | 5 | -5, 2000 | 1,1560 | Text, e.g. ‘x’ |
| 36 | Test entry of end position in file for Stenography | 100 | -100, 2000 | 1,1560 | Text, e.g. ‘x’ |
| 51 | Test shift key values for the Caesar cipher | 10 | 0 | -25, -1 and 1, 25 | Text, e.g. ‘x’ |
| 83 | Test number of lines in rail fence encryption | 5 | -8, number greater than plaintext length | 1, number of characters in plain text | Text, e.g. ‘x’ |
| 124 | Test character position ‘jump’ value (N) in file for Stenography | 10 | -10, 2000 | 1, number of characters | Text, e.g. ‘x’ |
| 188 | Test continue | Enter key | Any other key | Enter Key | N/A |

1. Complete the table to list all of the places that the code where a validation test is either missing OR incomplete (i.e. the user could enter invalid data causing an error)

|  |  |  |
| --- | --- | --- |
| **Line number of input** | **Validation Type missing** | **Description of Validation inc. criteria NEEDED to be added (include code if you can)** |
| 31 | Presence Check | Get text (plain text and cipher text). Test that text has been entered by checking that the length of the string is greater than zero.  valid = False  while not valid:  TextFromUser = input('Please enter the text to use: ')  if len(TextFromUser) < 1:  print('You must enter some text!')  else:  valid = True |
| 35 | Type check  Range check | Test entry of start position in file for Stenography.  Use try/except to handle runtime error caused by character input (e.g. ‘x’).  Apply range check if int conversion succeeds.  valid = False  while not valid:  try:  StartPosition = int(input('Please enter the start position to use in the file: '))  # Type check has succeeded so now do range check  if StartPosition < 1 or StartPosition > 1560: # range check  print('Start position must be between 1 and 1560')  else:  valid = True  except:  print('You must enter a numberic value!')  Notes:   1. better to get length of the file than hard-code 1560 2. could use isdigit() function to test for numeric input but it doesn’t handle negative numbers |
| 36 | ^ | Test entry of end position in file for Stenography.  Similar to above. Note that range check print statement uses f’{variableName}’ syntax.  valid = False  while not valid:  try:  EndPosition = int(input('Please enter the end position to use in the file: '))  # Type check has succeeded so now do range check  if EndPosition < StartPosition or EndPosition > 1560:  print(f'End position must be between {StartPosition} and 1560')  else:  valid = True  except:  print('You must enter a numeric value!') |
| 51 | ^ | Test shift key values for the Caesar cipher  Note that isdigit() not a good option here as it doesn’t handle negative numbers:  valid = False  while not valid:  try:  Key = int(input('Enter the amount that shifts the plaintext alphabet to the ciphertext alphabet: '))  if Key < -25 or Key == 0 or Key > 25 : # range check  print('You must enter a number between -25 and -1 or between 1 and 25')  else:  valid = True  except:  print('You must enter a numeric value!') |
| 83 | ^ | Test number of lines in rail fence encryption  Check that number of rail fences is not larger than the number of characters in the plain text.  valid = False  while not valid:  try:  SizeOfRailFence = int(input('Enter the number of lines in the rail fence: '))  if SizeOfRailFence < 1 or SizeOfRailFence > len(Plaintext) : # range check  print(f'You must enter a number between 1 and {len(Plaintext)} (length of the Plain Text)')  else:  valid = True  except:  print('You must enter a numeric value!') |
| 124 | ^ | Test character position ‘jump’ value (N) in file for Stenography  Use isdigit() function to implement type check:  Check that jump is not larger than the difference between the start and end positions.  valid = False  while not valid:  try:  N = int(input('Enter the value of n: '))  if N < 1 or N > (EndPosition - StartPosition) : # range check  print(f'You must enter a number between 1 and the difference between the start and end positions ({EndPosition-StartPosition})')  else:  valid = True  except:  print('You must enter a numeric value!') |

1. Complete the table to list all of the assignment statements in the code where processing occurs, try to explain/describe what is happening to the data during the execution of that line(s) (feel free to put a few lines together.

A selection of assignmenrt statements…

|  |  |  |
| --- | --- | --- |
| **Line Number** | **Assignment Statement** | **Explanation** |
| 46 | TextFromFile+=ByteFromFile.decode('ASCII') | Having read the last byte from the file, convert it to an ASCII string character and append it to the TextFromFile string variable. |
| 67 | NewASCIICode = ((26 + ASCIICode - ord('A') + AmountToShift) % 26) + ord('A') | Applies the Caeser shift by the specified amount for upper case characters. Apply Modulus 26 to handle shift beyond ‘Z’ or before ‘A’. |
| 69 | NewASCIICode = ((26 + ASCIICode - ord('a') + AmountToShift) % 26) + ord('a') | Applies the Caeser shift by the specified amount for lower case characters. Apply Modulus 26 to handle shift beyond ‘z’ or before ‘a’. |
| 77 | ASCIICode = ord(OriginalText[Count]) | Converts each character of the input text into ASCII code to be shifted |
| 79 | ChangedText += chr(ASCIICode) | Creates the encrypted or decrypted text as each character of the input text is shifted using the ApplyShiftToASCIICodeForCharacter function |
| 91 | Ciphertext += OriginalText[Count2] | In RailFence encryption, adds the character from the position stored in count2 (see next description) |
| 92 | Count2=Count2+SizeOfRailFence | Increments count2 by the number of ‘fences’ in the rail fence to determine the position of the next character to add to the ciphertext. |
| 131 | HiddenMessage += GetTextFromFile(CurrentPosition, CurrentPosition) | Gets character at position ‘CurrentPosition’ from the text file and concatenates it to the HiddenMessage variable |
| 132 | CurrentPosition = CurrentPosition + N | Moves current position along by the input jump value, N. |
| 155 | Plaintext = GetTextFromUser() | The variable Plaintext is set to the return from the GetTextFromUser function |
| 159 | Ciphertext = GetTextFromUser() | ^ |
| 164 | AmountToShift = GetKeyForCaesarCipher() | The variable amount to shift is set to the result from GetKeyForCeaserCipher() |
| 165 | Ciphertext = UseCaesarCipher(Plaintext, AmountToShift) | The variable Ciphertext is set to the return from UseCeaserCipher() with the parameters Plaintext and amount to shift. |
|  |  |  |