# Assessment test

1. (a) Give **three** essential properties of a recursive routine. [3]

(b) Which of the following subroutines are recursive? [3]

(i) SUB addNumbers(num,x)

IF num > 1

x = num + addNumbers(num – 1, x)

ENDIF

RETURN x

ENDSUB

(ii) SUB printNumbers(num)

OUTPUT (num)

RETURN num + printNumbers(num – 1)

ENDIF

ENDSUB

(iii) SUB numbers(num)

IF num <= 0 THEN

RETURN 0

ELSE

RETURN numbers(num) - 1

ENDIF

ENDSUB

(c) The following pseudocode subroutine merges two sorted lists of equal or unequal lengths. It is called with the two lists shown in the main program.

NOTE: Given that list1 = [1, 2, 5, 10], then list1[1:] = [2, 5, 10].

That is, it returns a sublist starting with list1[1].

1. SUB merge (list1,list2)
2. IF length(list1) = 0 and length(list2) = 0
3. return []
4. ENDIF
5. IF length (list1) = 0
6. return list2
7. ENDIF
8. IF length (list2) = 0
9. return list1
10. ENDIF
11. IF list1[0] <= list2[0]
12. return [list1[0]] + merge(list1[1:], list2)
13. ENDIF
14. IF list2[0] <= list1[0]
15. return [list2[0]] + merge(list1, list2[1:])
16. ENDIF
17. ENDSUB
18. #main program
19. list1 = [3, 7, 9]
20. list2 = [1, 2, 5, 10]
21. mergedList = merge(list1,list2)
22. print(mergedList)

(i) What will be printed when the statement print(mergedList) at line 23 is executed? [1]

(ii) Give the statement numbers of the recursive calls in the subroutine. [2]

(iii) What parameters will be passed to the subroutine the first, second and third times that a recursive call is made? [3]

First time:

Second time:

Third time:

2. (a) The Big-O notation gives a measure of the time complexity of an algorithm relative   
 to the size of the problem.

Three different algorithms A, B, C to solve the same problem are:

A = O(n2), B = O(log2n) and C = O(n!)

(i) Which is the most efficient algorithm in terms of execution time? [1]

(ii) Which is the least efficient algorithm in terms of execution time? [1]

(b) Use the Big-O notation to express the complexity of the following pseudocode segments. In each case, show how you arrive at the answer.

(i) i = n

WHILE i > 0

i = i DIV 2 # (DIV =integer division)

ENDWHILE [2]

(ii) FOR i = 1 TO n

FOR j = 1 TO n

FOR k = 1 TO n

listX[i, j, k] = 0

ENDFOR

ENDFOR

ENDFOR [2]

(c) A subroutine big(aList) is given below.

SUB bigO(aList)

numberOfPrints = 0

n = length(aList)

FOR i = 0 TO n-1

OUTPUT("In outer loop: ", aList[i])

numberOfPrints = numberOfPrints + 1

k = n/2 - 1

FOR j = 0 TO k

OUTPUT(" In inner loop: ",aList[j])

numberOfPrints = numberOfPrints + 1

ENDFOR

ENDFOR

OUTPUT("number Of Print statements executed: ", numberOfPrints)

numberOfPrints = numberOfPrints + 1

ENDSUB

#main program

listOfItems = [1,2,3,4,5,6]

bigO(listOfItems)

(i) Calculate the number of times an OUTPUT statement is executed in the subroutine **bigO(aList),** shown below, called from the main program and accepting as a parameter a list of 6 items. [4]

(ii) Hence, express its time complexity using Big-O notation and explain your answer.

[2]

3. A procedure to process an array of numbers is defined as follows.

SUB p(listA)

i 🡨 length(listA)-1

flag 🡨 True

WHILE i > 0 AND flag = True

flag 🡨 False

FOR j = 0 to i - 1

IF listA[j] > listA[j + 1] THEN temp 🡨 listA(x)

temp = listA[j]

listA[j] 🡨 listA[j + 1]

listA[j + 1] 🡨 temp

flag 🡨 True

ENDIF

ENDFOR

i 🡨 i - 1

ENDWHILE

ENDSUB

The array listA[19,10,21,7,26,16] is to be processed by this procedure.

(a) List the array after the WHILE loop has been executed once. [2]

(b) What algorithm does the procedure p describe? [1]

(c) What is the purpose of flag in this procedure? [2]

(d) (i) Trace the execution of the subroutine p if

listA = [2, 5, 8, 12, 16] [5]

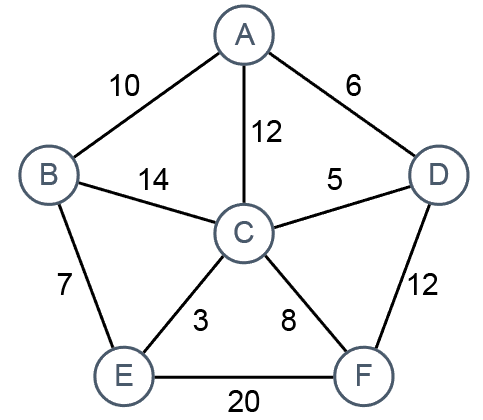
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **i** | **Flag** | **j** | **listA[j]** | **listA[j + 1]** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

(ii) How many times is the WHILE loop executed? [1]

4. Djikstra’s algorithm finds the shortest path between a given start node and every other node in a weighted graph.

(a) Describe briefly **two** applications of Djikstra’s algorithm. In each case state what the weights on the edges represent. [4]

Shown below is a weighted graph.



(b) In the algorithm below to find the shortest distances from A to the other nodes, a tentative distance from A to every other node is initially assigned.

(i) In the table below, show what these tentative distances are. [2]

(ii) In the second row of the table, show the tentative distances after A and the next node have been visited. [4]

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | | **C** | | **D** | | **E** | | **F** | |
| Initial tentative distance |  | |  | |  | |  | |  | |  | |
| tentative distance after next node visited |  | |  | |  | |  | |  | |  | |

Assign a tentative distance value to every node

Add all the vertices to a priority queue, sorted by current distance

WHILE priority queue is not empty

remove the vertex u from the front of the queue

FOR each unvisited neighbour w of the current vertex u

newDistance 🡨 distanceAtU + distanceFromUtoW

IF newDistance < distanceAtW THEN

distanceAtW 🡨 newDistance

change position of w in priority queue to reflect new distance to w ENDIF

ENDFOR

ENDWHILE

5. (a) What is meant when a problem is said to be **intractable**? [2]

(b) Give an example of an intractable problem. [1]

(c) Name and describe briefly another approach to solving a problem if no algorithm can be found which will find a solution in a reasonable time. [2]

Total 50 marks