Exam-style Questions

1 **a**

```
FUNCTION FindName(s : STRING) RETURNS INTEGER
   Index = -1
   First ← 0
   Last ← 50
   WHILE (Last >= First) AND Index = -1
      Middle ← (First + Last) DIV 2
      IF Names[Middle] = s
         THEN
            Index ← Middle
         ELSE
            IF Names[Middle] < s
               THEN
                  Last ← Middle + 1
               ELSE
                  First ← Middle - 1
            ENDIF
      ENDIF
   ENDWHILE
ENDFUNCTION
```

Marking guidance:

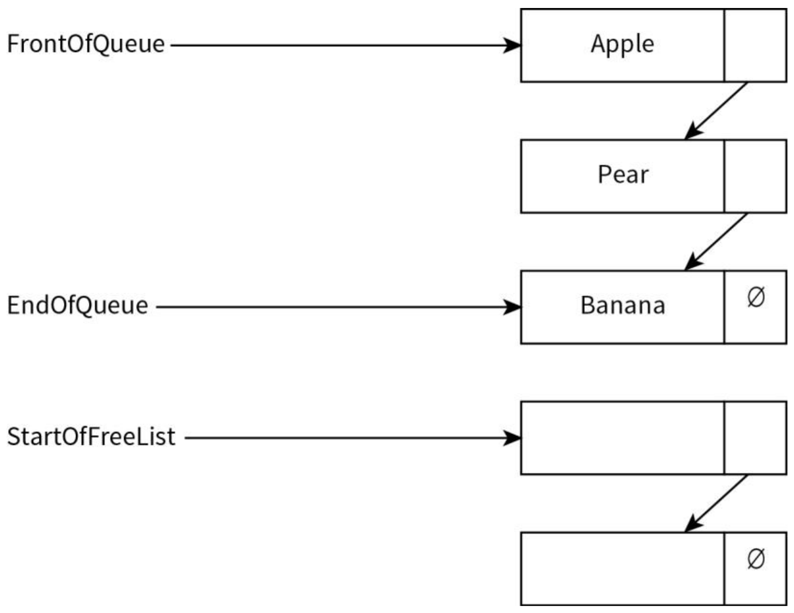
1 mark for each correct line of pseudocode completed

b the binary search does not work if the data in the array being searched is not sorted in ascending order. [1 mark]

c **i** the function returns the index of the search item [1 mark]

ii the function returns -1 [1 mark]

2 **a** **i, ii**



Marking guidance part i:

1 mark for correct three data items in three different nodes (don't have to be in first three boxes)

1 mark for data items linked together in the order Apple, Pear, Banana

1 mark for null pointer in node of Banana

1 mark for FrontOfQueue pointing to Apple

1 mark for EndOfQueue pointing to Banana

Marking guidance part ii:

1 mark for StartOfFreeList pointing to an empty node

1 mark for all empty nodes linked together with null pointer in final node

b i

Python	`class Node :`
	`   def __init__(self) :`
	`      self.Data = ""`
	`      self.Pointer = -1`

Marking guidance:

1 mark for correct record header and end

1 mark for each correct field with suitable data type

ii

Python	`Queue = [Node() for i in range(50)]`
	
	`FrontOfQueue = -1`
	`EndOfQueue = -1`
	`StartOfFreeList = 0`
	`for i in range(49) :`
	`   Queue[i].Pointer = i + 1`

Marking guidance:

1 mark for correct declaration of Node array

1 mark for initialisation of FrontOfQueue and EndOfQueue

1 mark for correct initialisation of StartOfFreeList

1 mark for loop addressing all nodes

1 mark for initialising pointer field of node within loop

1 mark for incrementing pointer value correctly

1 mark for null pointer in final node pointer field

c i

Identifier	Data Type	Description
`NullPointer`	`INTEGER`	Constant set to -1
`Queue`	`Node`	Array to store queue data
`NewItem`	`STRING`	Value to be added
`StartOfFreeList`	`INTEGER`	Pointer to next free node in array
`FrontOfQueue`	`INTEGER`	Pointer to first node in queue
`EndOfQueue`	`INTEGER`	Pointer to last node in queue
`NewNodePointer`	`INTEGER`	Pointer to node to be added

Marking guidance:

1 mark for NewItem

1 mark for Queue

1 mark for correct data type for Queue

1 mark for StartOfFreeList with suitable data type

1 mark for FrontOfQueue with suitable data type

1 mark for EndOfQueue with suitable data type

1 mark for NewNodePointer with suitable data type

ii
```
PROCEDURE JoinQueue(NewItem : STRING)
   // Report error if no free nodes remaining
   IF StartOfFreeList = NullPointer
      THEN
         Report Error
      ELSE
         // new data item placed in node at start of free list
         NewNodePointer ← StartOfFreeList
         Queue[NewNodePointer].Data ← NewItem
         // adjust free list pointer
         StartOfFreeList ← Queue[NewNodePointer].Pointer
         Queue[NewNodePointer].Pointer ← NullPointer
         // if first item in queue then adjust front of queue pointer
         IF FrontOfQueue = NullPointer
            THEN
               FrontOfQueue ← NewNodePointer
         ENDIF
         // new node is new end of queue
         Queue[EndOfQueue].Pointer ← NewNodePointer
         EndOfQueue ← NewNodePointer
   ENDIF
ENDPROCEDURE
```

Marking guidance:

1 mark for each correctly completed gap in the pseudocode

3 **Python**

```
MAXITEMS = 500

def OutputList(WordList, NumberOfWords):
# Q 3b
   for Index in range(NumberOfWords):
      print(WordList[Index])


def LoadWords(WordList):
# Q 3c
   FileName = input("Which file do you want to use? ")
   NumberOfWords = 0
   Index = 0
   try:
      FileHandle = open(FileName, "r")
      Word = FileHandle.readline()
      WordList[Index] = Word.rstrip('\n') # remove
line feed
      while len(Word) > 0 and Index < MAXITEMS:
         Index += 1
         Word = FileHandle.readline()
         WordList[Index] = Word.rstrip('\n') # remove
line feed
         if Index == MAXITEMS - 1:
            print("List full")
      FileHandle.close()
      NumberOfWords = Index + 1
   except:
      print("File not found")
   return (WordList, NumberOfWords)

def SortWordList(WordList, NumberOfWords):
# Q 3d
   n = NumberOfWords - 2
   NoMoreSwaps = False
   while NoMoreSwaps == False:
      NoMoreSwaps = True
      for j in range(n):
         if WordList[j] > WordList[j + 1]:
            Temp = WordList[j]
            WordList[j] = WordList[j + 1]
            WordList[j + 1] = Temp
            NoMoreSwaps = False
      n = n - 1
   return (WordList)


def main():
# Q 3e
   WordList = ["" for i in range(MAXITEMS)]
# Q 3a
```

```
    WordList, NumberOfWords = LoadWords(WordList)
    print ("unsorted list: ")
    OutputList(WordList, NumberOfWords)
    print("======================================")
    WordList = SortWordList(WordList, NumberOfWords)
    print ("sorted list: ")
    OutputList(WordList, NumberOfWords)

main()
```

Marking guidance for part a:

1 mark for declaration of array WordList with 500 elements

1 mark for initialising every element

1 mark for setting initial value to emptystring

Marking guidance for part b:

1 mark for procedure heading with correct parameters

1 mark for loop addressing every word in the list

1 mark for print statement within the loop

Marking guidance for part c:

1 mark for procedure heading

1 mark for asking user for file name

1 mark for opening file for reading

1 mark for reading each line in file and assigning it to the next array element

1 mark for closing the file

1 mark for exception handling in case of non-existent file

1 mark for returning updated WordList and NumberOfWords (or use reference parameters or global variables)

Marking guidance for part d:

1 mark for procedure heading

1 mark for declaring local variables

1 mark for setting up outer loop correctly

1 mark for each correctly completed gap in the pseudocode

1 mark for setting up inner loop correctly

1 mark for correct IF statement within inner loop

1 mark for correctly swapping elements

Marking guidance for part e:

1 mark for calling LoadWords at the beginning of the main program

1 mark for calling OutputList after LoadWords

1 mark for calling SortWords followed by OutputList

Marking guidance for part f:

1 mark for test run showing trying to use a non-existing file

1 mark for test run showing the list of words, first unordered, and then ordered.