1. A group of students are designing a racing car game. The game will allow players to enter their name and then a choice of vehicle. They will then race against other vehicles that will be controlled by the program. Players will use the arrow keys to control their vehicle.

The students are identifying the inputs and outputs for the game.

Complete the table by identifying **two** inputs and **two** outputs for the game.

Input 1	....
Input 2	....
Output 1	....
Output 2	....

[4]

2. A program is being designed that will allow a user to log into an account on a website using a username and password.

Identify **two** possible inputs and **one** output this program will need.

Input 1 _____

_____

Input 2 _____

_____

Output _____

[3]

3(a). A programmer has designed a program that includes a reusable program component.

The reusable program component is a function called `isInteger()`. This will take a string as an argument and then check that each digit is between 0 and 9. For example if 103 is input, it will check that the digits 1, 0 and 3 are each between 0 and 9.

The `asc()` function returns the ASCII value of each digit. For example `asc("1")` returns 49.

The ASCII value for 0 is 48. The ASCII value for 9 is 57.

```
01    function isInteger(number)
02       result = true
03       for count = 0 to number.length-1
04          asciiValue = asc(number.substring(count, 1))
05             if not(asciiValue >= 48 and asciiValue <= 57) then
06             result = false
07          endif
08       next count
09       return result
10    endfunction
```

i. Identify **one** identifier used in the function `isInteger()`.

---

[1]

ii. Give the line number where the branching (selection) construct starts in the function `isInteger()`.

---

[1]

iii. Give the line number where the iteration construct starts in the function `isInteger()`.

---

[1]

(b). Give **two** reasons why reusable program components are used in programs.

1 _____

_____

2 _____

---

[2]

4. Taylor is designing a program for a client who would like to simulate earthquakes on major cities around the world in 3D. The client would like to be able to view any stage of an earthquake such as:

1. the build-up of the earthquake
2. the earthquake taking place
3. the aftershocks of the earthquake.

The client would also like to be able to play the simulation at different speeds. For example, a slow, normal or fast speed.

The program will need to accept inputs from the user before playing the simulation.

i. Identify **two** different inputs for this program.

1 ---

---

2 ---

--- **[2]**

ii. One decision point in the program will be to decide if the user inputs are suitable or not.

Identify **two** other example decision points in this program.

1 ---

---

2 ---

--- **[2]**

5. A programmer has partially developed a bubble sort algorithm in pseudocode.

This will partially sort an array of numbers called `numbers` that is passed as a parameter.

```
01 procedure bubbleSort(numbers : byRef)
02   flag = true
03   for x = 0 to numbers.length - 1
04      if numbers[x] > numbers[x + 1] then
05         holdValue = numbers[x]
06         numbers[x] = numbers[x + 1]
07         numbers[x + 1] = holdValue
08         flag = false
09      endif
```

```
10    next x
11 endprocedure
```

i. Explain why the procedure `bubbleSort` accepts the array `numbers` by reference and not by value.

[3]

ii. The programmer has used a `for` loop on line 3 in the procedure `bubbleSort`. A `for` loop is a count controlled loop.

State what is meant by the term 'count controlled loop'.

[1]

iii. State the purpose of the variable `holdValue` in the procedure `bubbleSort`.

[3]

iv. The procedure `bubbleSort` will only partially sort the array `numbers` into order.

Describe what the programmer would need to add to the algorithm to enable it to fully sort the numbers into order.

[2]

6. Lucas writes a program that makes use of a circular queue. The queue stores the data entered into the program. An array is used to represent the queue.

The program needs two pointers to access and manipulate the data in the queue.

State the purpose of the two pointers and give an appropriate identifier for each.

Pointer 1 purpose

Pointer 1 identifier

Pointer 2 purpose

Pointer 2 identifier

---

[4]

7(a). Hugh has written a recursive function called `thisFunction()` using pseudocode.

```
01 function thisFunction(theArray, num1, num2, num3)
02   result = num1 + ((num2 - num1) DIV 2)
03   if num2 < num1 then
04     return -1
05   else
06     if theArray[result] < num3 then
07        return thisFunction(theArray, result + 1, num2, num3)
08     elseif theArray[result] > num3 then
09        return thisFunction(theArray, num1, result - 1, num3)
10     else
11        return result
12     endif
13   endif
14 endfunction
```

The function `DIV` calculates integer division, e.g. 5 DIV 3 = 1

`theArray` has the following data:

Index:	0	1	2	3	4	5	6	7
Data:	5	10	15	20	25	30	35	40

Trace the algorithm, and give the final return value, when it is called with the following statement:

```
thisFunction(theArray, 0, 7, 35)
```

You may choose to use the table below to give your answer.

Function call	num1	num2	num3	result
`thisFunction(theArray,0,7,35)`				

Final return value ..... **[5]**

(b). State the name of the standard algorithm `thisFunction()` performs.

---

[1]

(c). Hugh could have written `thisFunction()` using iteration instead of recursion.

Compare **two** differences between recursion and iteration.

1

2

[4]

(d). The recursive function `thisFunction()` is printed again here for your reference.

```
01 function thisFunction(theArray, num1, num2, num3)
02   result = num1 + ((num2 - num1) DIV 2)
03   if num2 < num1 then
04     return -1
05   else
06     if theArray[result] < num3 then
07        return thisFunction(theArray, result + 1, num2, num3)
08     elseif theArray[result] > num3 then
09        return thisFunction(theArray, num1, result - 1, num3)
10     else
11        return result
12     endif
13   endif
14 endfunction
```

Rewrite the function `thisFunction()` so that it uses iteration instead of recursion.

You should write your answer using pseudocode or program code.

[6]

8. The following pseudocode procedure performs an insertion sort on the array parameter.

```
01 procedure insertionSort(dataArray:byRef)
02   for i = 1 to dataArray.Length - 1
03     temp = dataArray[i]
04     tempPos = i - 1
05     exit = false
06     while tempPos >= 0 and exit == false
07         if dataArray[tempPos] < temp then
08             dataArray[tempPos + 1] = dataArray[tempPos]
09             tempPos = tempPos - 1
10         else
11             exit = true
12         endif
13     endwhile
14     dataArray[tempPos + 1] = temp
15   next i
16 endprocedure
```

State whether the procedure `insertionSort` sorts the data into ascending or descending order and explain your choice.

[3]

9(a). A printer buffer is a storage area that holds the data, known as jobs, that are to be printed by a printer.

A simulation of the printer buffer uses a queue data structure to store jobs that are waiting to be printed. The queue is not circular.

The printer buffer is represented as a zero-indexed 1D array with the identifier `buffer`.

Fig. 2 shows the current contents of the queue `buffer` and its pointers.

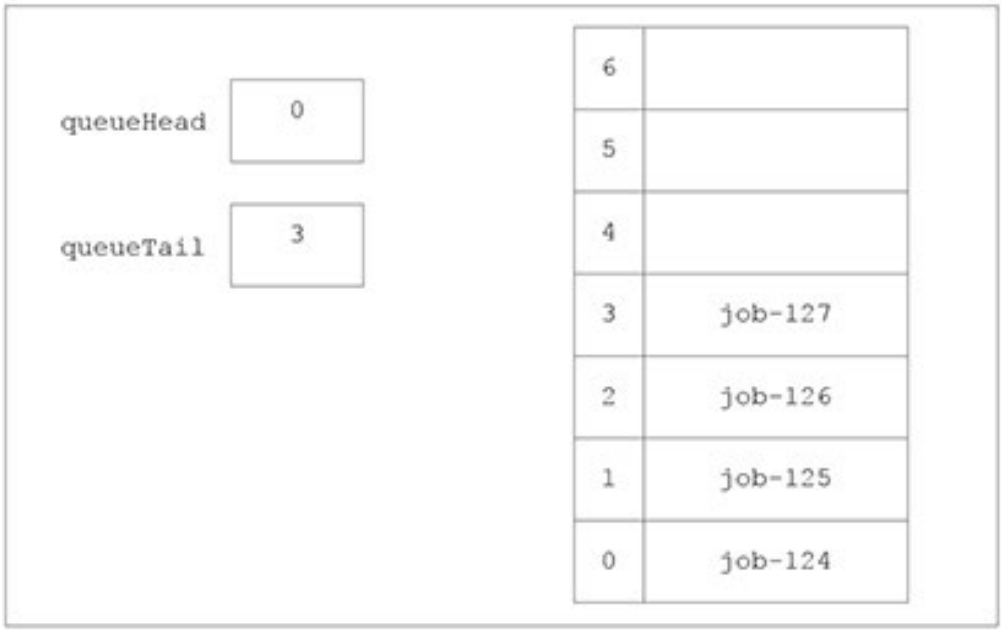


Fig. 2

State the purpose of the pointers `queueHead` and `queueTail`.

`queueHead` _____

_____

`queueTail` _____

_____

[2]

(b). The function `dequeue` outputs and removes the next data item in the queue.

The procedure `enqueue` adds the job passed as a parameter to the queue.

Show the final contents of the queue and pointer values after the following instructions have been run on the queue `buffer` shown in **Fig. 2**.

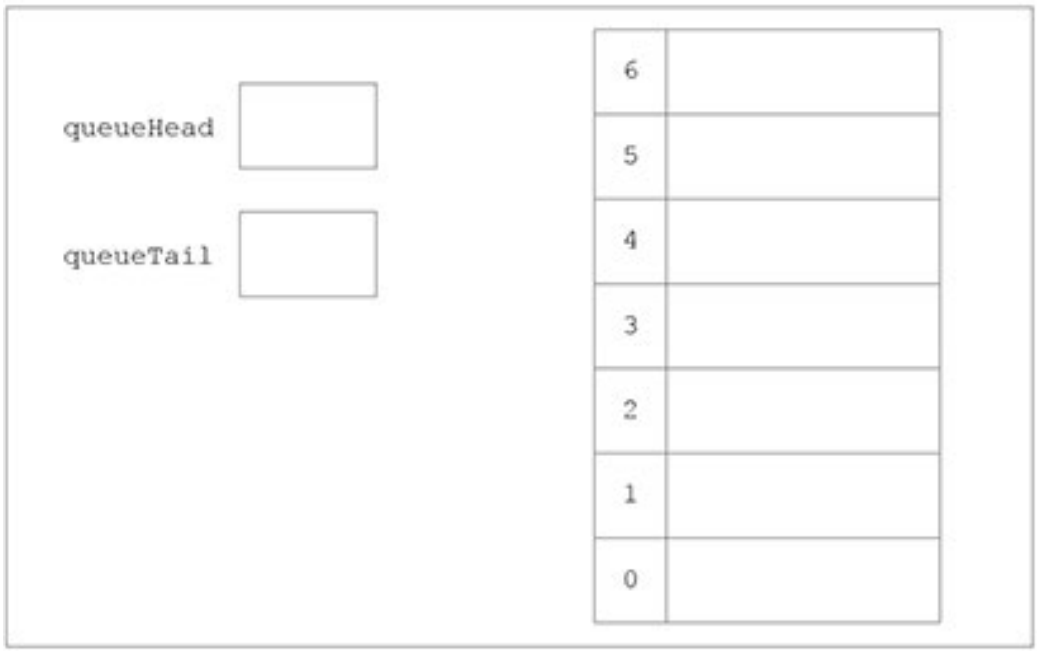
```
dequeue()

dequeue()

enqueue(job-128)

dequeue()

enqueue(job-129)
```



[5]

(c). The array, `buffer` and pointer values are declared with global scope.

i. The function `dequeue` returns `null` if the array is empty, and the contents of the next element if not empty. The queue is not circular.

Write an algorithm, using pseudocode or program code, for the function `dequeue()`.

[5]

ii. The function `enqueue` returns -1 if there is no space at the end of the queue to add data, and returns 1 if the parameter was added to `buffer`. The array `buffer` contains a maximum of 100 elements.

Write an algorithm, using pseudocode or program code, for the function `enqueue()`.

[6]

iii. In the main program of the simulation the user is asked whether they want to add an item to the queue or remove an item.

If they choose to add an item they have to input the job name, and the function `enqueue` is called.

If they choose to remove an item, the function `dequeue` is called and the job name is output.

Appropriate messages are output if either action cannot be run because the queue is either empty or full.

Write, using pseudocode or program code, an algorithm for the main program of the simulation.

[8]

(d). The queue is changed to make it a circular queue.

Describe how the functions `enqueue` and `dequeue` will need to be changed to allow `buffer` to work as a circular queue.

[3]

(e). Some print jobs can have different priorities. The higher the priority the sooner the job needs to be printed.

Describe how the program could be changed to deal with different priorities.

[3]

10(a). Given the following procedure:

```
procedure maths(number)
    a = (number DIV 10) * 10
    b = a + 10
    if (number - a) >= (b - number) then
       print(b)
    else
       print(a)
    endif
endprocedure
```

State the value printed by the procedure `maths` if `number=10` _____ **[1]**

(b).

State the value printed by the procedure `maths` if `number=27` _____ **[1]**

(c).

State the value printed by the procedure `maths` if `number=14` _____ **[1]**

END OF QUESTION PAPER