GCST

2.3.1 ALGORITHMS FOR THE MAIN DATA STRUCTURES

TOPIC WISE EXAM QUESTIONS

A-LEVEL OCR

GCSECOMPUTERSCIENCETUTOR.COM

1 A tree is one example of a data structure.

(a) (i) Give **two** characteristics of a tree data structure.

1 ....

....

2 ....

....
[2]

(ii) The following data is entered into a binary search tree.

22 13 5 36 55 14 8

Draw the binary search tree when the given data is entered in the order given.

[4]

(iii) Describe how a **leaf node** is deleted from a binary search tree.

....

....

....

.... [2]

(iv) Describe how a binary search tree can be searched for a value.

....

....

....

....

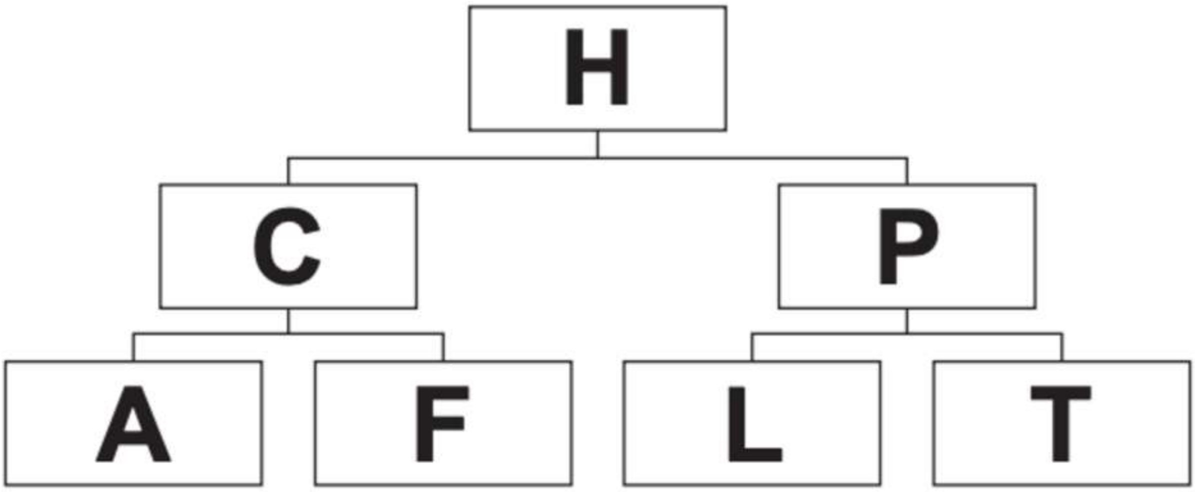
....

....

....

.... [4]

(v) Identify the order that the nodes will be visited in a **depth-first (post-order)** traversal of this binary search tree.



.... [4]

(vi) Explain how backtracking is used in depth-first (post-order) traversals.

....

....

....

.... [2]

3 A program stores data in a linked list.

The current contents of the linked list are shown in **Fig. 3**, along with the linked list pointers.

`headPointer`	1
`freeListPointer`	4

location	data	pointer
0	"blue"	6
1	"red"	0
2	"green"	8
3	"orange"	NULL
4		5
5		7
6	"grey"	2
7		9
8	"purple"	3
9		NULL

Fig. 3

(a) State the purpose of `headPointer` and `freeListPointer` in the linked list shown in **Fig. 3**.

`headPointer` ....

....

`freeListPointer` ....

....

[2]

(b) State the meaning of the pointers with the value `NULL` in the linked list shown in **Fig. 3**.

....

.... **[1]**

(c) A procedure outputs the data in the linked list shown in **Fig. 3** from the first item in the list, to the last item.

Give the output from the procedure.

....

.... **[2]**

(d) A new item needs to be added to the linked list.

Describe how a new item is added to a linked list.

....

....

....

....

....

....

....

.... **[4]**

(e) The function `findNode` will search the linked list and return either the position of the node that contains the data item, or -1 if the data item is not found.

The data held in a node at location `x` can be accessed with `linkedList[x].data`. The pointer of the node at location `x` can be accessed with `linkedList[x].pointer`.

For example, using the linked list shown in **Fig. 3**:
`linkedList[2].data` returns `green`.
`linkedList[2].pointer` returns `8`.

Complete the function, using pseudocode or program code.

```
function findNode(toFind, headPointer, linkedList)
  currentNode = ....................................
  while(currentNode != ....................................)
    if linkedList[currentNode]. .................................... == toFind then
      return currentNode
    else
      currentNode = linkedList[....................................].pointer
    endif
  endwhile
  return ....................................
endfunction
```

[5]

2 A computer program is being written to store data about students.

Fig. 2 shows a binary search tree that stores data about students.

Each student is represented by their ID number. The current contents of the binary search tree are:

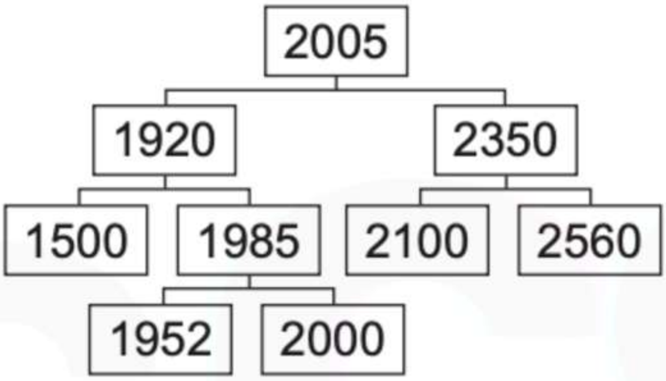


Fig. 2

(a) Identify the root node in the binary tree shown in **Fig. 2**.

....

.... **[1]**

(b) Identify **two** leaf nodes in the binary tree shown in **Fig. 2**.

1 ....

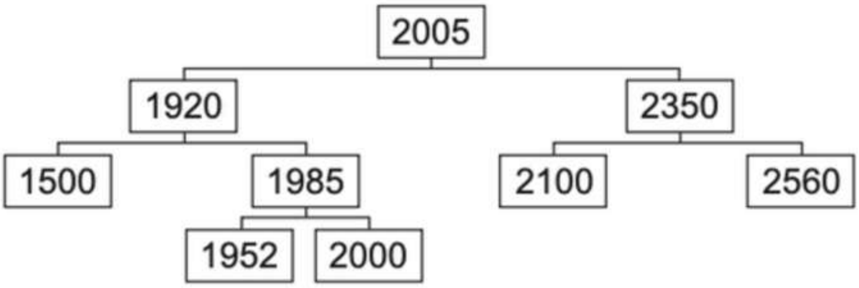
2 ....

[2]

(c) Four more students are added to the binary search tree shown in **Fig. 2** in this order:

1420 2050 2780 2600

Complete the binary search tree here by adding the new students to it.



[4]

(d)* A programmer would like to traverse the binary search tree shown in **Fig. 2**.

Compare the use of a breadth-first traversal and a depth-first (post-order) traversal on the binary search tree.

You should include the following in your answer:

- how each traversal works
- the order of the return values for each traversal. [9]

....

....

....

....

....

....

....

....

....

....

5 **Fig. 5** shows a graph data structure representing a small section of a parcel delivery network. Each node represents an address where deliveries need to be made. The edges show the possible routes and distances between these deliveries.

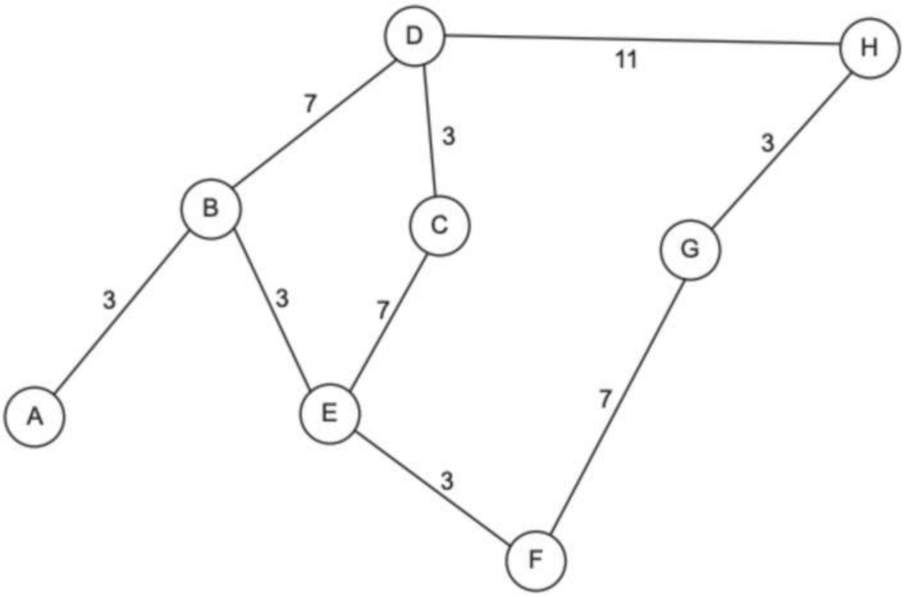


Fig. 5

(a) **(i)** State **one** way a directed graph is different to an undirected graph.

....

.... **[1]**

(ii) State **one** way a graph data structure is different to a tree data structure.

....

.... **[1]**

8 A computer uses a stack data structure, implemented using an array, to store numbers entered by the user.

The array is zero based and has 100 locations.

(a) **Fig. 8** shows the current contents of the stack and the first 9 locations of the array.

pointerValue	5

Index	Data
8	
7	
6	
5	
4	1
3	23
2	6
1	5
0	10

Fig. 8

(i) The function `pop()` removes an item from the stack.

The function `push()` adds an item to the stack that is passed in as a parameter.

Show the contents of the stack and pointer from **Fig. 8** after the following subroutines calls have run.

```
pop()

pop()

push(3)

push(6)

push(7)
```

pointerValue	

Index	Data
8	
7	
6	
5	
4	
3	
2	
1	
0	

[2]

(ii) State the purpose of `pointerValue`.

....

.... **[1]**

(b) The stack is programmed as an object using object-oriented programming. The design for the class, its attributes and methods are shown:

```
class: stack
---------------------------------------
attributes:
private stackArray : Array of integer
private pointerValue : integer
---------------------------------------
methods:
new()
function pop()
function push(value)
```

(i) The method `pop()` returns the next value in the stack, or `-1` if the stack is empty.

Complete the pseudocode method `pop()`.

```
public function pop()
  if pointerValue == ................................................ then
    return ................................................
  else
    pointerValue  = pointerValue ................................................
    returnValue = stackArray[................................................]
    return ................................................
  endif
endfunction
```

[5]

(ii) The method `push()` accepts an integer as a parameter and adds it to the top of the stack unless the stack is already full.

If the push is successful the method returns `true`.

If the push is unsuccessful due to the stack being full the method returns `false`.

Write the method `push()` using either pseudocode or program code.

....

....

....

....

....

....

....

....

....

....

....

....

....

.... **[6]**

(iv) The main program needs to:

- take numbers as input from the user
- push them onto the stack `mathsStack` until the stack is full
- output an appropriate message if the stack is full.

Complete the pseudocode algorithm to meet these requirements.

```
returnValue = true

while returnValue == ..............................................

    returnValue = mathsStack. ........................................(input("Enter
    Number"))

    if returnValue == ............................................... then

        .............................................. ("Stack full")

    endif

endwhile
```

[4]

(v) The main program also needs to:

- remove one item from the stack at a time and add this to a total
- output the total every time an item is removed
- stop removing items when either the stack is empty, or 20 items have been removed.

Write pseudocode or program code to meet these requirements. [8]

....

....

....

....

....

....

....

....

....

....

....

(c) The program is amended to include the use of several queue data structures.

(i) Describe how an array can be used to implement a queue data structure.

....

....

....

....

....

.... [3]

(ii)* Discuss the use of object-oriented programming and procedural programming to create and manipulate the queue data structures.

You should include the following in your answer:

- the features of object-oriented programming
- the features of procedural programming
- the benefits of using object-oriented instead of procedural programming when creating several queue structures. [9]

....

....

....

....

....

....

....

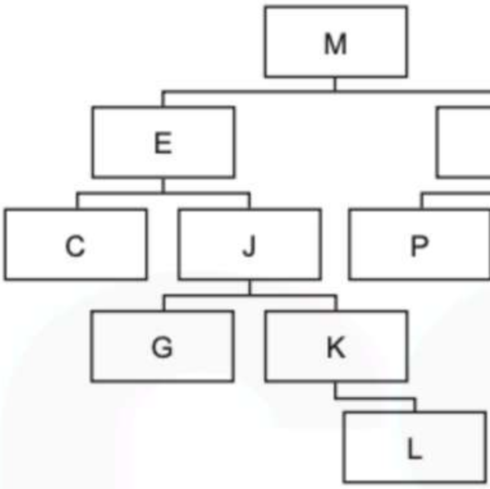
....

....

....

1 **(d)** A breadth-first traversal can be performed on both a tree and a graph.

Show how a breadth-first traversal is performed on the following binary tree.



....

....

....

....

.... **[6]**

7 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.

(a) 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]

(b) Lucas wants a procedure, `enqueue()`, that will add the parameter it is passed to the queue.

Describe the steps the procedure `enqueue()` will follow when adding new items to the queue.

....

....

....

....

....

....

....

....

....

.... [5]

1 Kira is creating a computer game where the user can play against the computer.

In each turn, each character can make one move from a selection of possible moves.

Kira uses a tree data structure shown in **Fig. 1** to identify the range of possible moves the computer can make from starting position A. Each connection is a move, with each node representing the result of the move.

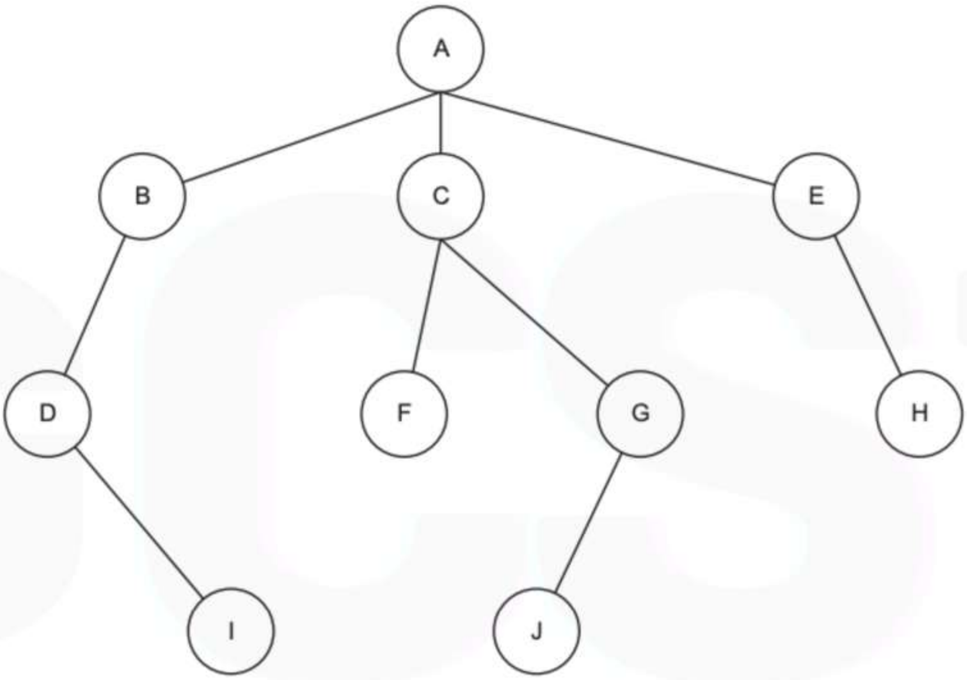


Fig. 1

(b) State why the tree shown in **Fig. 1** is **not** an example of a binary search tree.

....

.... **[1]**

(c) State what type of pointers are used to store nodes I, F, J and H so they do not point to any other nodes.

....

.... **[1]**

Kira wants the program to traverse the tree to evaluate the range of possible moves. She is considering using a breadth-first traversal or a depth-first (post-order) traversal.

(d) Show how a breadth-first traversal would traverse the tree shown in **Fig. 1**.

....

....

....

....

....

....

....

.... **[4]**

(e) Kira wants to make some changes to the data that is stored in the tree structure shown in **Fig. 1**.

(i) The move represented by node 'E' needs to be deleted.

Describe the steps an algorithm will follow to delete node 'E' from the tree.

....

....

....

....

....

.... **[3]**

(ii) The move represented by the node 'K' needs to be added. Node 'K' needs to be joined to node 'G.'

Describe the steps the algorithm will follow to add node 'K' to the right of node 'G'.

....

....

....

....

....

.... **[3]**

(f) Kira could have used a graph data structure to represent the moves in her game.

Give **two** similarities and **two** differences between a tree and a graph data structure.

Similarity 1 ....

....

Similarity 2 ....

....

Difference 1 ....

....

Difference 2 ....

....

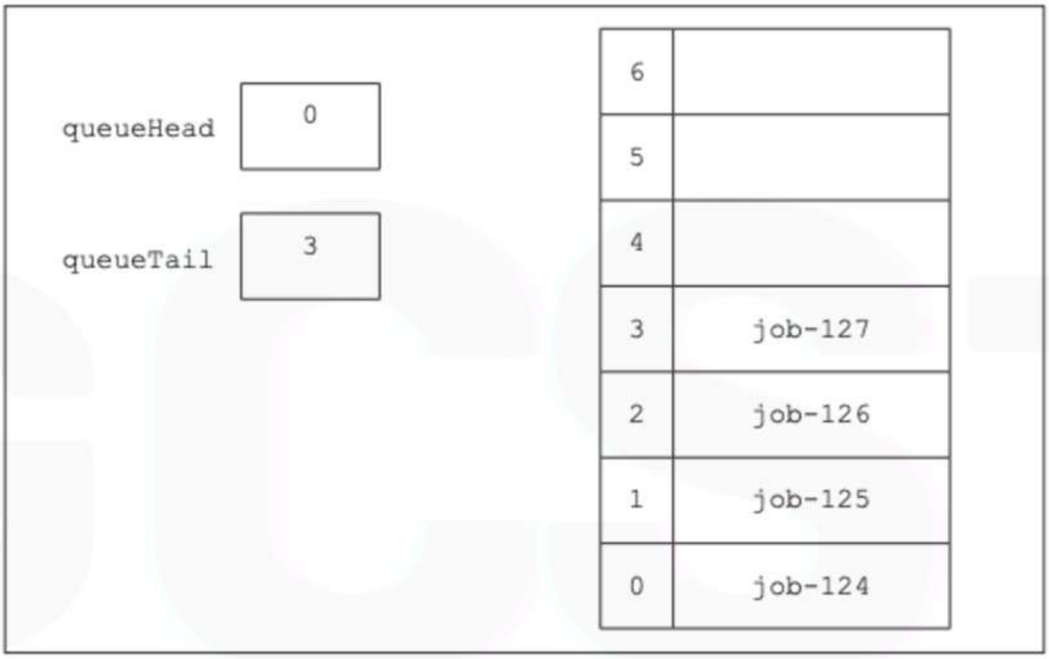
[4]

5 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.



queueHead 0

queueTail 3

6	
5	
4	
3	job-127
2	job-126
1	job-125
0	job-124

Fig. 2

(a) 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)
```

queueHead	
queueTail	

6	
5	
4	
3	
2	
1	
0	

[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]**

6 Barney is writing a program to store data in a linked list. He is writing the initial program for a maximum of 10 data items.

Each node in the linked list has a data value and a pointer (to the next item).

A null pointer is stored with the value `-1`.

(a) **Fig. 3** shows the current contents of the linked list including the head and free list pointer values.

`headPointer` 0

`freeListPointer` 4

index	data	pointer
0	2.6	3
1	3.5	-1
2	1.8	1
3	6.9	2
4		5
5		6
6		7
7		8
8		9
9		-1

Fig. 3

(i) Describe the purpose of `freeListPointer`.

....

....

....

.... **[2]**

(ii) State the purpose of `headPointer`.

....

.... **[1]**

(iii) Show the contents of the linked list from **Fig. 3** and the pointer values when the node with data `6.9` is deleted.

`headPointer`

`freeListPointer`

index	data	pointer
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

[4]

(b) Barney wants the nodes to be stored as objects using object-oriented programming. He designs the following class.

```
class:     node
----------------------------------
attributes:
private   data : Real
private   pointer : Integer
----------------------------------
methods:
new (newData, newPointer)
getData()
getPointer()
setData(newData)
setPointer(newPointer)
```

(c) The function `findNodePath()` takes the data item to find in the linked list as a parameter and follows the pointers to find the required node.

The function returns the array indexes of all the nodes it visits and joins this to a suitable message stating whether the data was found or not found and then returns this as one string.

Describe how the function `findNodePath()` will search for the data item and return the required message.

....

....

....

....

....

....

....

....

....

....

....

.... **[6]**

(d) The procedure `printLinkedList()` follows the pointers to print all of the elements in the linked list.

```
01 procedure printLinkedList(headPointer)
02   tempPointer = headPointer - 1
03   dataToPrint = ""
04   if tempPointer == -1 then
05     print("List is full")
06   else
07     while linkedList[pointer].getPointer() != -1
08       dataToPrint = dataToPrint + " " + linkedList[tempPointer,0]
09        linkedList[tempPointer].getPointer() = tempPointer
10     endwhile
11    print(dataToPrint + " " + linkedList[tempPointer].getData()
12   endif
13 endprocedure
```

The procedure has a number of errors.

(i) Identify the line of each error and write the corrected line.

Error 1 line number ....

Error 1 correction ....

Error 2 line number ....

Error 2 correction ....

Error 3 line number ....

Error 3 correction ....

[3]

(e)* Barney would like his linked list to be part of a base program that is saved in a library. This means that it can be reused and changed by other programs.

Discuss the benefits of using different object-oriented techniques that Barney could use to achieve this. [12]

3 Oscar owns a taxi company. He would like a program to handle taxi bookings from customers.

(a) When a customer makes a booking, they are placed into a queue data structure until a taxi driver is available.

(i) Explain why Oscar uses a queue data structure rather than a stack data structure.

....

....

....

....

....

....

....

.... [4]

(ii) Oscar has written a procedure, `enqueue`, to be able to add a customer number to the queue. The queue is not circular.

```
01  procedure enqueue(custNumber)
02     maxElements = 10
03     if (tail + 1) > maxElements then
04        print ("Error, queue is full")
05     else
06        head = head + 1
07        queue[head] = custNumber
08     endif
09  endprocedure
```

State the name of the parameter used in the procedure `enqueue`.

....

.... [1]

(iii) The procedure `enqueue` contains an error on line 06 and line 07.

Rewrite lines 06 and 07 of the procedure `enqueue` so that the queue works correctly.

....

....

....

.... [2]

(iv) Identify the logical condition in the procedure `enqueue` that affects whether a new Item can be added to the queue.

....

.... [1]

1 The temperatures of an ocean are input into a computer system. They are recorded, and will be accessed, in the order in which they arrive. The data for one week is shown:

5, 5.5, 5, 6, 7, 6.5, 6

(a) The data is to be stored in a data structure. The programmer stores the data in a queue.

Explain why a queue is used instead of a stack.

....

....

....

.... **[2]**

(b) The data is processed. After processing, the value for the first day is stored as 0. The value for each following day is stored as an increase, or decrease, from the first day.

For example: if the first day was 7, the second was 6 and the third was 9, after processing it would be stored as 0, –1, 2.

(i) The queue uses `dequeue()` to return the first element of the queue.

`dequeue()` is a function.

Explain why `dequeue()` is a function, not a procedure.

....

.... **[1]**

(ii) Complete the algorithm to process the data in the queue and store the results in an array called `processedData`.

```
processedData[0] = 0
firstDay = ..............................
for count = 1 to 6
    processedData[..........................] = dequeue() - ..........................
next count
```

[3]

2 A program needs to store the names of plants that are in a garden, so they can be easily found and accessed in alphabetical order.

The data is stored in a tree structure. Part of the tree is shown.

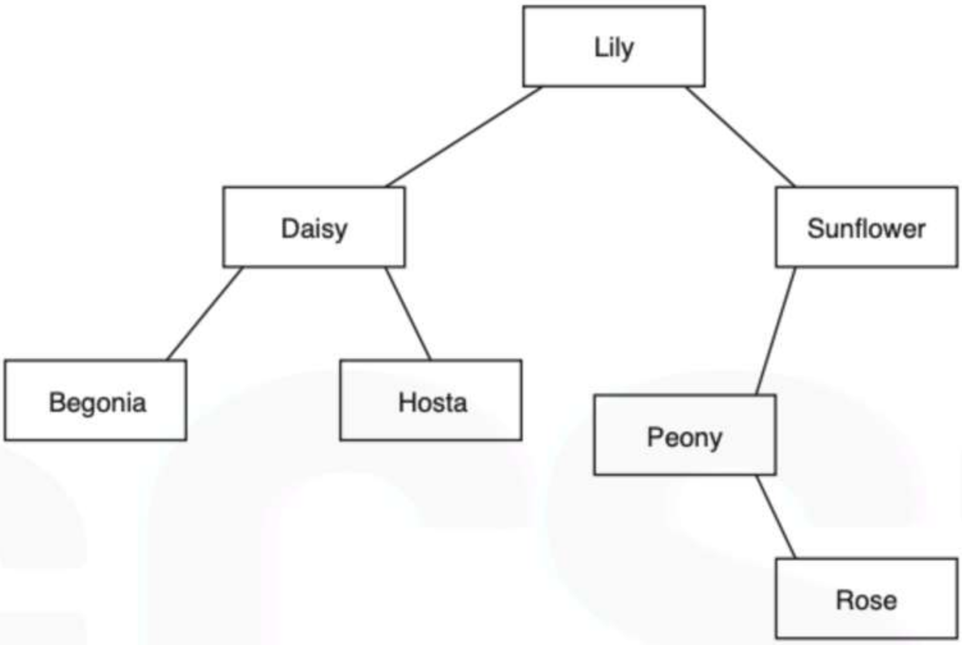


Fig. 2.1

(a) **(i)** State the type of tree shown in Fig. 2.1.

.... **[1]**

(ii) Show the output of a breadth-first traversal of the tree shown in Fig. 2.1.

....

....

....

....

....

....

....

.... **[3]**

(iii) Explain how backtracking is used in a depth-first (post-order) traversal. Use the tree in Fig. 2.1 in your explanation.

....

....

....

....

....

....

....

....

....

....

....**[4]**

(b) The elements in the tree in Fig. 2.1 are read into a linked list producing an alphabetised list.

(i) Complete the following table to show the linked list for the data.

Data item	Data	`NextPointer`
0	Begonia	
1	Daisy	
2	Hosta	
3	Lily	
4	Peony	
5	Rose	
6	Sunflower	

[2]

(ii) A new plant, Lavender, needs adding to the linked list. The linked list needs to retain its alphabetical order.

Complete the table to show the linked list after Lavender is added.

Data item	Data	NextPointer
0	Begonia	
1	Daisy	
2	Hosta	
3	Lily	
4	Peony	
5	Rose	
6	Sunflower	

[3]

(iii) Hosta needs removing from the linked list.

Explain how a data item is removed from a linked list. Use the removal of Hosta in your answer.

....

....

....

....

....

....

....

.... [4]

(iv) The linked list is stored as a 2D array with the identifier `plantList`. The index of the first element of the linked list is stored in the identifier `firstElement`.

All contents of the linked list need to be output in alphabetical order.

Write an algorithm to follow the pointers to output the contents of the linked list in alphabetical order.

Add comments to explain your code.

....

....

....

....

....

....

....

....

....

....

....

.... **[5]**

3 The current contents of a queue, `colours`, implemented in an array is shown in Fig. 3.1.

red	yellow	green	blue	grey			

```
front = 0

end = 4
```

Fig. 3.1

(a) Describe the purpose of `front` and `end`.

....

....

....

.... **[2]**

(b) The queue has the subprograms `enqueue` and `dequeue`. The subprogram `enqueue` is used to add items to the queue and the subprogram `dequeue` removes items from the queue.

(i) Use the following diagram to show the queue shown in Fig. 3.1 after the following program statements have run:

```
enqueue("orange")
dequeue()
enqueue("maroon")
dequeue()
dequeue()
```


`front =` ....

`end =` ....

[4]

(ii) `enqueue` and `dequeue` are both functions.

State the difference between a procedure and a function.

....

.... **[1]**

(iii) Describe the steps involved in the `enqueue` algorithm.

....

....

....

....

....

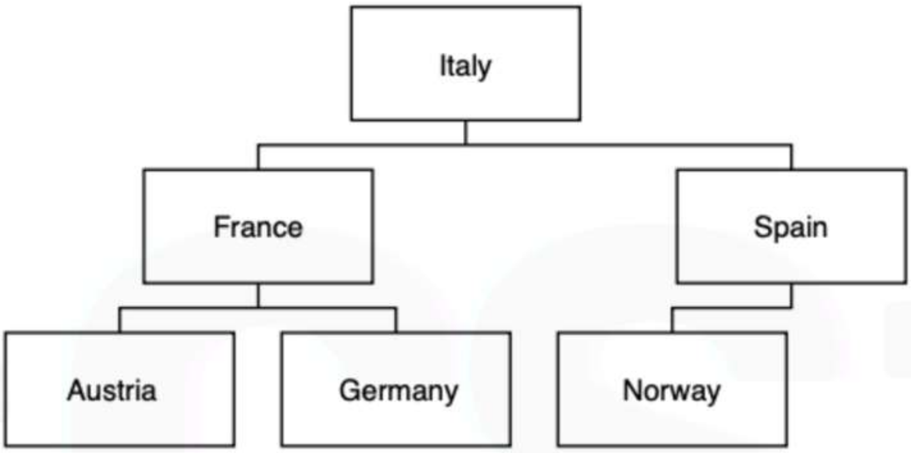
....

....

.... **[4]**

1 A program stores entered data in a binary search tree.

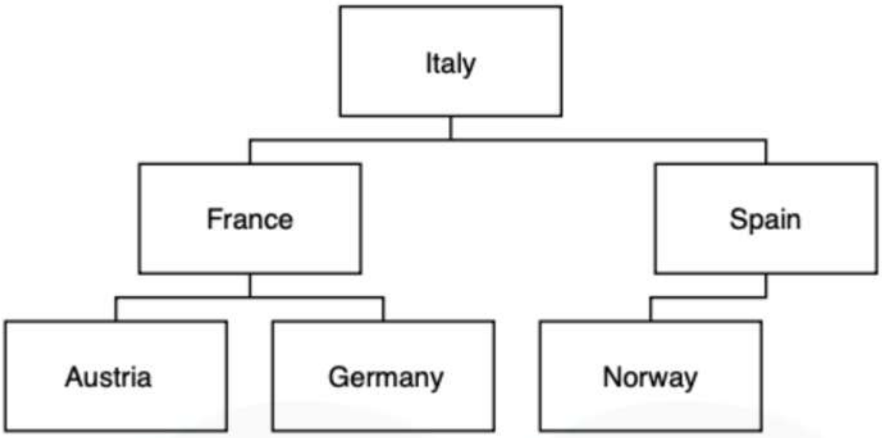
The current contents of the tree are shown:



(a) Complete the diagram to show the contents of the tree after the following data is added:

England, Scotland, Wales, Australia **[3]**

(b) Show the order of the nodes visited in a breadth first traversal on the following tree.



....

....

....

....

....

....

....

....**[3]**

(c) A pseudocode algorithm is written to search the tree to determine if the data item "Sweden" is in the tree.

The function `currentNode.left()` returns the node positioned to the left of `currentNode`.

The function `currentNode.right()` returns the node positioned to the right of `currentNode`.

```
function searchForData(currentNode:byVal, searchValue:byVal)
    thisNode = getData(..................................................)
    if thisNode == .............................................. then
        return ..............................................
    elseif thisNode < searchValue then
        if currentNode.left() != null then
            return (searchForData(currentNode.left(), searchValue))
        else
            return ..............................................
        endif
    else
        if .............................................. != null then
            return (searchForData(currentNode.right(), searchValue))
        else
            return false
        endif
    endif
endfunction
```

(i) Complete the algorithm.

[5]

(ii) The algorithm needs to be used in different scenarios, with a range of different trees.

Identify **two** preconditions needed of a tree for this algorithm to work.

1 ....

2 ....

[2]

5 A computer program stores data input on a stack named `dataItems`. The stack has two sub-programs to add and remove data items from the stack. The stack is implemented as a 1D array, `dataArray`.

Sub-program	Description
`push()`	The parameter is added to the top of the stack
`pop()`	The element at the top of the stack is removed

The current contents of `dataItems` are shown:

6
15
100
23

(a) Show the contents of the stack `dataItems` after each line of the following lines of code are run

```
01  push(13)
02  pop()
03  push(10)
04  push(20)
```

Line 01	Line 02	Line 03	Line 04
6			
15			
100			
23			

[4]

(b) The main program asks a user to push or pop an item from the stack. If the user chooses 'push', the data item is added to the stack. If the user chooses "pop", the next item is removed from the stack, multiplied by 3 and output.

The main program is shown:

```
01 userAnswer = input("Would you like to push or pop an item?")
02 if userAnswer == "push" then
03     push(input("Enter data item"))
04 else
05     print(pop() * 3)
06 endif
```

(i) Before the sub-programs, `push()` and `pop()`, can add or remove items from the stack, a selection statement is used to decide if each action is possible.

Describe the decision that needs to be made in each sub-program and how this impacts the next process.

`push()` ....

....

....

....

`pop()` ....

....

....

....

[4]

(ii) The algorithm does not work when the user enters `"PUSH"` or `"Push"`. The algorithm needs to be changed in order to accept these inputs.

Identify the line number to be changed and state the change that should be made.

Line number ....

Change..

....

[2]

(c) The stack is implemented as a 1D array, `dataArray`.

Describe how a 1D array can be set up and used to push and pop items as a stack.

....

....

....

....

....

.... **[3]**

6 Kamran is writing a program to manipulate the data for a set of items.

For each item, the program needs to store:

- Item name (e.g. Box)
- Cost (e.g. 22.58)
- Date of arrival (e.g. 1/5/2018)
- Transferred (e.g. true)

The items are added to a queue for processing.

The queue is defined as a class, `itemQueue`.

itemQueue
theItems[10] : Items head : Integer tail : Integer numItems : Integer

constructor
enqueuer()
dequeuer()
setnumItems()
getnumItems()

The `head` attribute points to the first element in the queue. The `tail` attribute points to the next available space in the queue. The `numItems` attribute states how many items are currently in the queue.

(a) The data about the items can be stored using either a record structure, or as objects of a class.

(i) Explain the similarities and differences between a record and a class.

....

....

....

....

....

.... **[3]**

(ii) Kamran chooses to use a record structure to store the data about the items.

Record structures may be declared using the following syntax:

```
recordStructure recordstructurename
    fieldname : datatype
    …
endRecordStructure
```

Complete the pseudocode to declare a record called `items`.

```
recordStructure ..........................................................................................

    itemName : .............................................................................

    ..............................................................................: Currency

    ..............................................................................: Date

    transferred : .............................................................................

endRecordStructure
```

[5]

(iii) New records may be created using the following syntax:

```
recordidentifier : recordstructurename
recordidentifier.fieldname = data
…
```

Write a programming statement to create a new item, using the identifier 'box1', with the item name "Box", the cost 22.58, date of arrival 1/5/2018 and transferred true.

....

....

....

....

....

....

....

.... **[3]**

(b) The array, `theItems`, stores the items in the queue. When the tail of the queue exceeds the last element in the array, it adds a new item to the first element if it is vacant.

For example, in the following queue, the next item to be added would be placed at index 0.

Index	0	1	2	3	4	5	6	7	8	9
Element				Data	Data	Data	Data	Data	Data	Data

(i) Define the term ‘queue’.

....

....

....

.... **[2]**

(ii) The attributes in `itemQueue` are all declared as private.

Explain how a private attribute improves the integrity of the data.

....

....

....

.... **[2]**

(iii) The constructor method creates a new instance of `itemQueue` and sets the `head`, `tail` and `numItems` attributes to 0.

Write an algorithm, using pseudocode or program code, for the constructor including the initialisation for all attributes.

....

....

....

....

....

.... **[2]**

(iv) The `enqueue` method:

- takes as a parameter the item to insert in the queue
- checks if the queue is full
- reports an error and returns `false` if the queue is full
- does the following if the queue is not full:
    - adds the item to the array at the tail position and adjusts the pointer(s)
    - returns `true`

The attribute `numItems` stores the number of items currently in the queue.

Write an algorithm, using pseudocode or program code, for the `enqueue` method.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... **[6]**

(v) Write a programming statement to declare an instance of `itemQueue` called `myItems`.

....

.... **[1]**

(vi) Write a procedure, `insertItems()`, to ask the user to input the data for an item. The item is then added to the queue `myItems`. The user is continually asked to input data items until the queue is full.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... **[5]**

1 A user enters whole numbers into a computer program. Each number entered is placed onto a stack. The stack is created using an array with a maximum of 20 elements.

Part of the array, `numStack`, is shown when one number has been input.

top	1

index	stackItem
9	
8	
7	
6	
5	
4	
3	
2	
1	
0	20

The pointer, `top`, points to the next free space in the stack.

(a) Complete the diagram below to show the state of `numStack` after the user inputs the following numbers in the order given:

22 13 2 59 1000

top	

index	stackItem
9	
8	
7	
6	
5	
4	
3	
2	
1	
0	20

[2]

(b) A function, `addItem`, takes a number as a parameter and adds the number to the stack. The function returns `true` if this was successful, and `false` if the stack is already full.

(i) Give **one** reason why a function is used instead of a procedure in this scenario.

....

.... **[1]**

(ii) The parameter can be passed by value or by reference.

Describe what is meant by passing a parameter by value and by reference.

By value ....

....

....

....

By reference ....

....

....

....

[4]

(iii) The function `addItem` is written but is incomplete.

Complete the function, `addItem`.

```
function addItem (number)
      if top == ........................................ then
            return false
      else
            numStack[........................................] = ....................................
            top = ........................................ + 1
            ..................................................................................
      endif
endfunction
```

[5]

(iv) The procedure, `calculate`, takes each item in turn from the stack. It alternately adds then subtracts the numbers until there are none left.

For example, if `numStack` contains:

2
6
5
12

It would perform 2 + 6 – 5 + 12 and output 15.

```
procedure calculate()
    total = 0
    add = true
    if top == 0 then
        print("Stack empty")
    else
        total = numStack[top - 1]
        top = top - 1
        while top != 0
            if add == true then
                total = total + numStack[top - 1]
                add = false
            else
                total = total - numStack[top - 1]
                add = true
            endif
            top = top - 1
        endwhile
        print(total)
    endif
endprocedure
```

Complete the trace table for the procedure `calculate`. The current array and pointer values when the procedure is called are on the first line of the trace table.

top	numStack						total	add	Output
	0	1	2	3	4	5			
5	20	2	6	12	8				

[6]

2 A games company has developed a game called Kidz Arrowz. The players throw an arrow at a target board and are awarded different points depending on which circle the arrow lands. Fig. 1 shows the board.

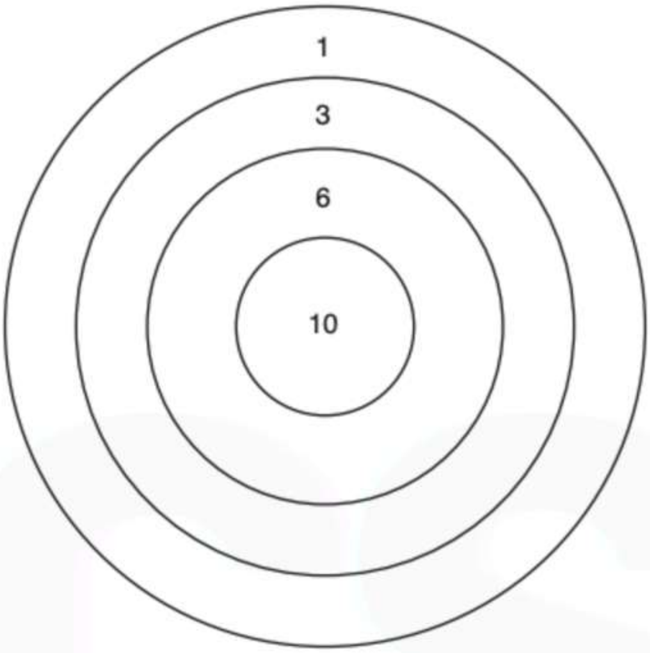


Fig. 1

A computer program is required to keep track of the scores for each competition. The user will enter the number of players, and the name of each player, in that competition to a maximum of 10. The program will then ask for the score of each player in turn. Each competition has 8 rounds, with each player throwing one arrow each round. The program will then display the total score of each player.

(a) **(i)** The players are declared as a record structure:

```
record player(string playerName, integer totalScore)
```

Describe what is meant by a record structure.

....

....

....

.... **[2]**

(ii) The records for the players are stored in a 1D array.

State why a 1D array is a suitable data structure for the records.

....

.... **[1]**

(iii) Three data structures are arrays, records and stacks.

Identify **one** other data structure.

.... **[1]**

(c) Player 1 is named Johnny. In the first round Johnny scores 3. Johnny can be added to the array using the code:

```
scores[0].playerName = "Johnny"
scores[0].totalScore = 3
```

(i) Write an algorithm to:

- allow the user to input and validate the number of players
- allow the user to input the name of each player
- output the round number at the start of a round
- display the player number for the score that needs to be entered
- allow the user to input the score for each player in that round and add it to their total.

[7]

....

....

....

....

....

....

....

....

....

....

....

....

2 A programmer is developing an ordering system for a fast food restaurant. When a member of staff inputs an order, it is added to a linked list for completion by the chefs.

(a) Explain why a linked list is being used for the ordering system.

....

....

....

.... **[2]**

(b) Each element in a linked list has:

- a pointer, `nodeNo`, which gives the number of that node
- the order number, `orderNo`
- a pointer, `next`, that points to the next node in the list

Fig. 2.1 shows the current contents of the linked list, `orders`.

nodeNo	orderNo	next
0	154	1
1	157	2
2	155	3
3	156	∅

Fig. 2.1

∅ represents a null pointer.

(i) Order 158 has been made, and needs adding to the end of the linked list.

Add the order, 158, to the linked list as shown in Fig. 2.1. Show the contents of the linked list in the following table.

nodeNo	orderNo	next

[2]

(ii) Order 159 has been made. This order has a high priority and needs to be the second order in the linked list.
Add the order, 159, to the original linked list as shown in Fig. 2.1. Show the contents of the linked list in the following table.

nodeNo	orderNo	next

[3]

(c) The linked list is implemented using a 2D array, `theOrders`:

- Row 0 stores `orderNo`
- Row 1 stores `next`

The data now stored in `theOrders` is shown in Fig. 2.2.

184	186	185	187
1	2	3	

Fig. 2.2

`theOrders[1,0]` would return `1`

The following algorithm is written:

```
procedure x()
    finished = false
    count = 0
    while NOT(finished)
        if theOrders[1,count] == null then
            finished = true
        else
            output = theOrders[0,count]
            print(output)
            count = theOrders[1,count]
        endif
    endwhile
    output = theOrders[0,count]
    print(output)

endprocedure
```

(i) Outline why `nodeNo` does not need to be stored in the array.

....

.... [1]

(iv) A new order, 190, is to be added to `theOrders`. It needs to be the third element in the list.
The current contents of the array are repeated here for reference:

184	186	185	187		
1	2	3			

Describe how the new order, 190, can be added to the array, so the linked list is read in the correct order, without rearranging the array elements.

....

....

....

....

....

....

....

.... [4]

3 An encryption routine reads a line of text from a file, reverses the order of the characters in the string and subtracts 10 from the ASCII value of each letter, then saves the new string into the same file.

The program is split into sub-procedures. Three sub-procedures are described as follows:

- Read string from file
- Push each character of the string onto a stack
- Read and encrypt each character message

(c) A function, `push`, can be used to add a character to a stack. For example:

```
theStack.push("H")
```

places the character `H` onto the stack, `theStack`.

A procedure, `pushToStack`, takes a string as a parameter and pushes each character of the message onto the stack, `messageStack`.

Complete the procedure below.

Add comments to explain how your code works.

```
procedure pushToStack(message)
```

....

....

....

....

....

....

....

....

```
endprocedure
```

[5]

(d) Describe the steps that the program would have to take in order to encrypt the characters stored in the stack, and save them in a single variable.

....

....

....

....

....

....

....

....

.... **[5]**

4 A data structure is shown below in Fig. 4.1.

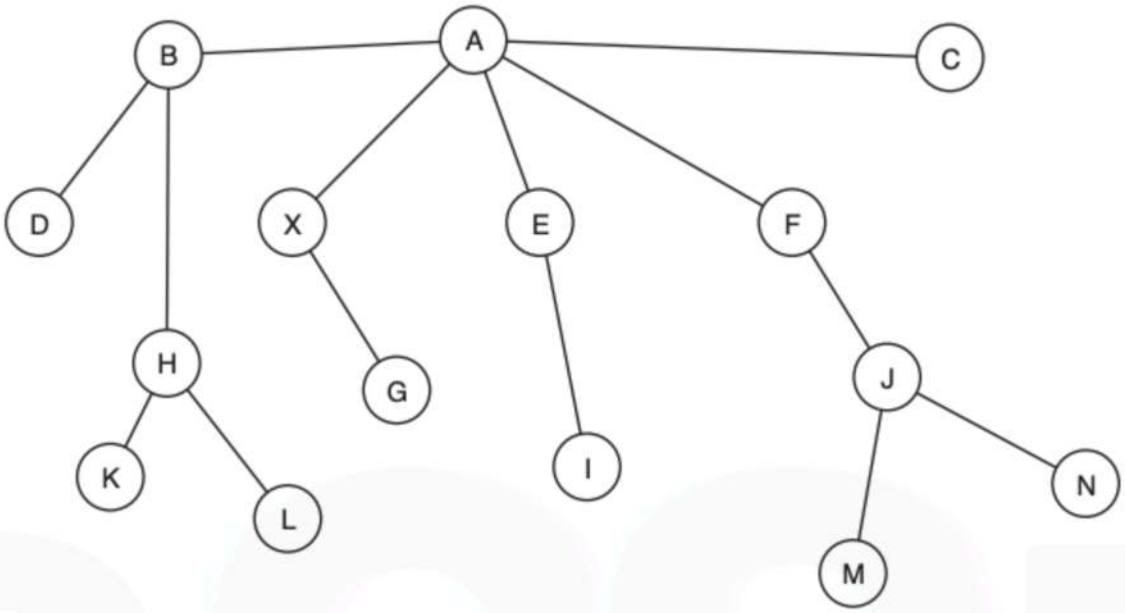


Fig. 4.1

(a) Identify the data structure shown in Fig. 4.1.

.... **[1]**

(b) The programmer is considering using a depth-first (post-order) traversal, or a breadth-first traversal to find the path between node A and node X.

(i) Explain the difference between a depth-first (post-order) and breadth-first traversal.

....

....

....

....

....

....

....

....

.... **[4]**

(ii) Show how a depth-first (post-order) traversal would find the path between node A and node X for the structure shown in Fig. 4.1.

....

....

....

....

....

....

....

....

....

....

....

....

....

.... **[6]**

(iii) Explain how you used backtracking in your answer to part **(b)(ii)**.

....

....

....

....

....

.... **[3]**

4 A program stores a queue of mathematical questions to be asked to a user. The questions are asked in the order they are added. Once a question has been asked it cannot be asked again. New questions are continually added to the end of the queue.

The program will use a non-circular queue, `questions`, (implemented using an array) to store the questions.
The pointer, `head`, stores the index of the first element in the queue.
The pointer, `tail`, stores the index of the last element in the queue.

(a) Describe why a queue is a suitable structure for this program.

....

....

....

....

....

.... **[3]**

(b) Fig. 4.1 shows an example of the data in the queue. `head` is currently 0, `tail` is currently 4.

"2*3"	"1+4"	"3-1"	"10/2"	"3+6"			

Fig. 4.1

(i) Show the contents of the queue shown in Fig. 4.1, after the following code is run.

`add("6+1")`

[2]

(ii) State the values stored in `head` and `tail` after the code in **part (i)** has run.

`head` ....

`tail` ....

[2]

(c) Complete the following algorithm, to remove, and output, the first element in the queue.

```
procedure remove()

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

endprocedure
```

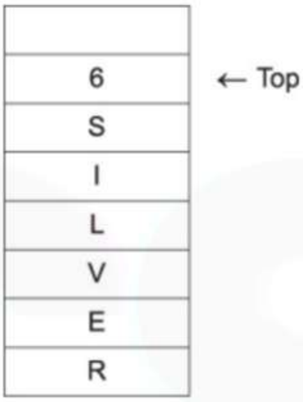
[4]

(d) Complete the following algorithm, to ask the user to input a new question and then either add it to the queue, or report that the queue is full.

```
procedure add()
    maxElements = 10

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

..............................................................................................................

endprocedure
```

[4]

4. A stack, in shared memory, is being used to pass a single variable length ASCII string between two sub-systems. The string is placed in the stack one character at a time in reverse order with the last byte holding the number of characters pushed i.e. the text "SILVER" would be held in the stack as:



Use pseudocode to write a procedure that will take a text string passed to it and push it to the stack in the format defined above. You may assume any given input will fit in the stack. [6]

If you found this useful, drop a follow to help me out!

THANK YOU!

GCST

GCSECOMPUTERSCIENCETUTOR.COM