GCST

1.4.2 DATA STRUCTURES

TOPIC WISE EXAM QUESTIONS

A-LEVEL OCR

GCSECOMPUTERSCIENCETUTOR.COM

GCST

SPEC

1.4.2 Data Structures	a) Arrays (of up to 3 dimensions), records, lists, tuples. b) The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table. c) How to create, traverse, add data to and remove data from the data structures mentioned above. *(This can be* ***either*** *using arrays and procedural programming or an object-oriented approach).*	Candidates should be able to describe what is meant by arrays (up to 3 dimensions), records, lists and tuples. They are expected to be able recognise when they can be used and incorporate them in their programs to store data. Candidates need to have an understanding of the behaviour of linked-lists, graphs, stacks, queues, trees, binary search trees and hash tables. Candidates need to be able be aware of how the aforementioned data structures can be implemented. We would recommend a general understanding of these principles that can be applied to a given scenario rather than trying to memorise code patterns. Candidates should have experience of implementing these structures in a variety of contexts, for example through a procedural program, through a different data structure and through an object-oriented approach. Candidates need to be able to read, trace and write code to implement features of these data structures. (Again we would recommend a general understanding backed up with practice implementing them, rather than trying to memorise code patterns).

2 Sundip writes an algorithm to carry out addition and subtraction. The algorithm will use an initially empty stack with the identifier `numbers` and will take input from the user.

The action the algorithm takes depends on the value input by the user. These actions are listed in **Fig. 2**.

Value input	Action to take
A	• Pop two values from the `numbers` stack • Add the two values • Push the result back onto the `numbers` stack
S	• Pop two values from the `numbers` stack • Subtract the first popped value from the second • Push the result back onto the `numbers` stack
E	• Pop one value from the `numbers` stack • Output this value • End program
Any other value	• Push the input value to the `numbers` stack

Fig. 2

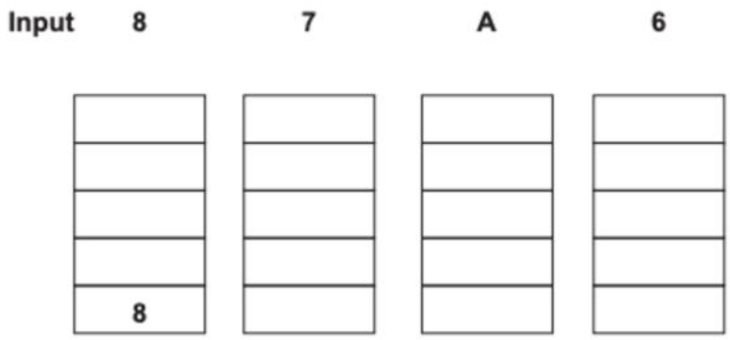
(a) Complete the pseudocode here to implement Sundip's algorithm.

```
do
    value = input("Enter a value")
    if .................................................... then
        num = numbers.pop()
        print(num)
    elseif value == "A" or ...................................... then
        numone = numbers.pop()
        numtwo = numbers.pop()
        if value == "A" then
            numbers.push.............................................
        elseif value == "S" then
            numbers.push(numtwo - numone)
        endif
    else
        numbers.push(................................)
    endif
until value == ...........................
```

[5]

(b) **(i)** Complete the diagram to show the state of the stack after each value is entered into the algorithm. The letters will complete an action stated in **Fig. 2**.

The state of the stack after the first value, 8, has been completed for you.



[3]

(ii) Complete the following table to give the output from this algorithm when the following set of inputs are entered by the user. The letters will complete an action stated in **Fig. 2**.

Input data (from left to right)	Output
9 3 A E	
10 5 A 8 S E	
25 5 S 2 3 A S E	

[3]

If the user enters **4 2 S A E**, the algorithm will not work correctly.

(iii) Explain what problem this input data will cause and why the problem occurs.

....

....

....

....

....

.... **[3]**

(c) A stack is one data structure that is available for Sundip to use. She could also use a queue, list, linked list, array or tuple.

(i) Describe **one** difference between a stack and a queue.

....

....

....

.... **[2]**

(ii) Describe **one** difference between an array and a list.

....

....

....

.... **[2]**

(iii) State **how a** tuple is different to a list.

....

.... **[1]**

(iv) Describe how the **second** item in a linked list would be accessed using pointer values.

....

....

....

....

....

.... **[3]**

2 A programmer uses a queue data structure to store data.

(a) **(i)** Tick **one** box that describes how a queue operates.

☐ Last In First Out

☐ First In First Out

[1]

(ii) The figure below shows a queue data structure that contains a list of names. Alex is at the front of the queue.

Alex	Kofi	Ben	Sundip	Tom			

The operations that can be used on the queue are:

- `enqueue()` – This will add data that is passed in as a parameter to the queue.
- `dequeue()` – This will return the first element in the queue.

Show the contents of the queue after these operations have been performed:

```
enqueue("Charlie")
dequeue()
enqueue("Ling")
dequeue()
enqueue("Sara")
```


[2]

(b) A stack is another type of data structure.

A stack is implemented using these variables:

- `items` – This is used to store an array that contains the data.
- `top` – This is an integer value pointing to the last item of data that was inserted.

`pop()` is one operation that can be performed on a stack. This will remove an item from the top of the stack, or –1 if the stack is empty.

(i) Complete the pseudocode function for the `pop()` operation.

```
function pop()
   if top == .................... then
      return -1
   else
      item = items[....................]
      top = top - ....................
      return ....................
   endif
endfunction
```

[4]

(ii) A function called `reverse` uses a stack called `theStack` to reverse data that is passed in as a parameter called `name`. For example, the `name` "Jack" would be returned as "kcaJ" by the function.

`theStack` uses these operations which are already defined as global scope in the program:

- `push()` – This will add data that is passed in as a parameter to the stack.
- `pop()` – This will remove and return the item on top of the stack.

Write the function `reverse` so that it:

- accepts the `name` as a parameter
- uses `push()` to add each character in the `name` to `theStack` separately
- uses `pop()` to return each character from `theStack` and add it to a variable called `reverseName`
- outputs the variable `reverseName` once all characters have been popped from `theStack`.

You should write your function using pseudocode or program code. [7]

7 A business uses an array with the identifier `wNames` to store workers' names. A variable with the identifier `top` is used to store the index of the last element to be added to the array, which is also the element which will next be removed.

`wNames`

0	1	2	3	4	5	6
Kirstie	Martyn	Louise	Alex	Anna		

`top`

4

(a) (i) State the name of the type of data structure described above.

.... **[1]**

(ii) Using pseudocode, write an algorithm that allows the user to enter a name which is then pushed onto the data structure above, checking first that the data structure is not full.

....

....

....

....

....

....

....

....

....

....

....

....

.... **[4]**

(b) The same workers' names are stored in a binary search tree which is ordered alphabetically.

Kirstie is set as the root node, with Martyn, Louise, Alex and Anna added one by one.

(i) Complete the tree diagram above to show where Martyn, Louise, Alex and Anna would be added to this binary search tree. **[4]**

(ii) Describe the process of using the binary search tree above to search for the name "Zoe".

[3]

(iii) Compare the efficiency of a binary search tree to a linked list when searching for data.

[2]

(iv) Compare the efficiency of a binary search tree to a hash table when searching for data.

[2]

5 A programmer is writing software for a firewall. She is writing code so that it keeps a track of websites that users are permitted to visit. The software stores the websites' addresses along with details about who can view them and when.

The following data is also stored about each website:

- Access level needed (`1-4`)
- If it is available all the time (`true`) or just lunch times and out of work hours (`false`).

So a website which is available to users of access level 2 and above, all the time, would have the details `[2, true]` stored.

A website accessible to users of access level 3 and above, only outside of work hours, would have the details `[3, false]` stored.

(a) State the name of a data structure that could be used to store a single site's details.

....

.... **[1]**

The address of each website, along with the relevant details, are stored in a hash table.

The hash table's hash function is carried out on the website's address (which acts as the key). The hash function works in the following way:

1. Discard the characters up to and including the first dot.
2. Discard the characters including and to the right of the remaining leftmost dot.
3. Convert the characters to uppercase.
4. Add the ASCII values of the characters together.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90

For example www.ocr.org.uk gets hashed in the following manner:

Step 1:

```
ocr.org.uk
```

Step 2:

```
ocr
```

Step 3:

```
OCR
```

Step 4:

```
79+67+82 = 228
```

giving a hashed value of 228.

(b) State what hashed value would be given by the website www.foo.co.uk

....

.... **[1]**

(c) Complete the function `hash` which takes in a string and returns the hashed value.

You can assume you have access to the following three functions.

- `asc()` – this takes in a character and returns its ASCII value. For example `asc("A")` returns 65.
- `locate()` – this takes in a string and character and returns the location of the first instance of the character (with the string starting at character 0). For example `locate("electricity","c")` returns 3.
- `upper()` – this takes in a string and returns the UPPERCASE version. For example `upper("hello")` returns "HELLO".

You should also assume that all given website names use letters but no numbers or symbols.

You will be given credit for the readability of your code.

```
function hash(siteName)









endfunction
```

[5]

A flaw with the current hash function is it tends to generate lots of collisions (addresses that compute to the same hash). Below is a diagram of part of the hash table. The address www.rnd.com with details `[2, true]` is being added to the hash table.

(d) Explain how a hash table can be used to handle collisions, referring to the example below.

227	
228	`www.ocr.org.uk : [1, true]`
229	
230	`www.ppf.nz : [2, false]`
231	
232	`www.ntf.biz : [4, true]`
234	
235	

.... [4]

The hash function is changed so there are no longer high numbers of collisions.

During busy periods the firewall is expected to check several addresses a second. It is anticipated that roughly 10 new addresses will be added to a whitelist (list of acceptable addresses) each day.

There is a debate as to whether a hash table (with the new hash function) is the best approach, or if the whitelist would be better stored in a linked list.

(e) *Discuss whether a hash table or linked list is better to store acceptable websites. You should compare how each structure can be searched and has data added and come to a recommendation as to which is better for the whitelist. **[12]**

4 Stacks and queues are both data structures.

(a) State which of a stack or queue would be considered as a 'First In First Out' data structure.

.... **[1]**

A stack is shown in Fig. 4.1 before a set of operations are carried out on it.

(b) Draw what the stack shown in Fig. 4.1 would look like after the following operations:

```
push("A"), push("B"), pop(), push("C"), pop(), push("D")
```

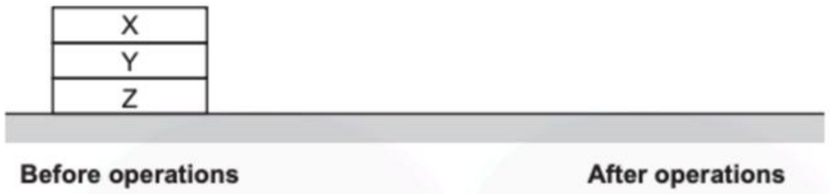


Fig. 4.1

[2]

Fig. 4.2 shows a stack in two states: State One and State Two.

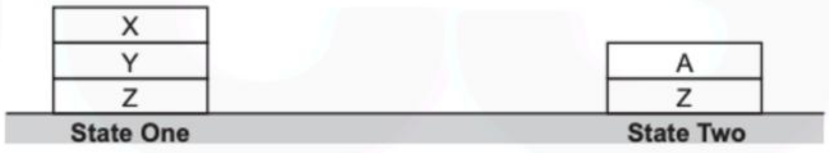


Fig. 4.2

(c) List the operations needed to get the stack from State One to State Two.

....

....

.... **[3]**

A queue is shown in Fig. 4.3.

(d) Draw what the queue shown in Fig 4.3 would look like after the following operations:

```
enqueue("A"), enqueue("B"), dequeue(), enqueue("C"), dequeue(),
enqueue("D")
```

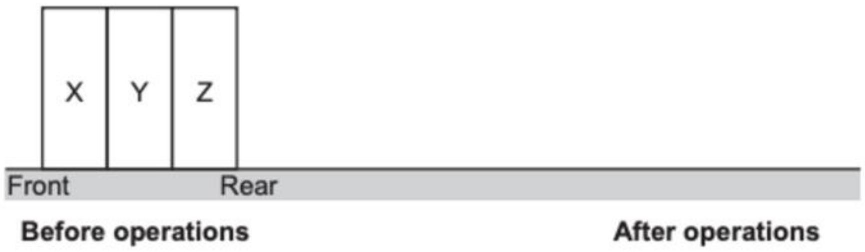


Fig. 4.3

[2]

6 A programmer has written the following code designed to take in ten names then print them in a numbered list.

```
name1 = input("Enter a name: ")
name2 = input("Enter a name: ")
name3 = input("Enter a name: ")
name4 = input("Enter a name: ")
name5 = input("Enter a name: ")
name6 = input("Enter a name: ")
name7 = input("Enter a name: ")
name8 = input("Enter a name: ")
name9 = input("Enter a name: ")
name10 = input("Enter a name: ")

print("1. " + name1)
print("2. " + name2)
print("3. " + name3)
print("4. " + name4)
print("5. " + name5)
print("6. " + name6)
print("7. " + name7)
print("8. " + name8)
print("9. " + name9)
print("10. " + name10)
```

It has been suggested that this code could be made more efficient and easier to maintain using an array or a list.

(a) Define the term 'array'.

....

....

....

.... **[2]**

(b) Write a more efficient version of the programmer's code using an array or a list.

....

....

....

....

....

....

....

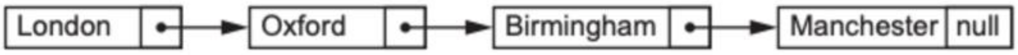
....

....

.... **[5]**

2 A coach company offers tours of the UK.

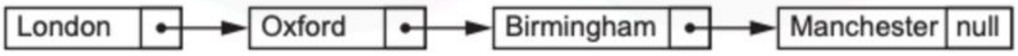
(a) A linked list stores the names of cities on a coach tour in the order they are visited.



(i) Describe what is meant by the term 'linked list'.

[3]

(ii) The tour is amended. The new itinerary is: London, Oxford, Manchester then York. Explain how Birmingham is removed from the linked list and how York is added. You may use the diagram below to illustrate your answer.



[4]

The program stores records about its customers.

(b) Often an individual customer's record needs to be accessed. This is done by searching using the Customer ID. Explain why a hash table is better suited than a linked list to store the customer records, particularly as the company acquires more customers.

....

....

....

....

....

....

....

....

....

.... **[4]**

7 A DIY store has an offer: 'Spend £20 or more on decorating products and get 10% off all gardening products.'

When items are scanned in at the checkout they are stored in a 2-dimensional array called `purchases`, which stores the item name, category and price.

A receipt with the appropriate discounts deducted is then produced.

Examples of the array and corresponding receipt are shown in Fig. 2 and Fig. 3.

Matt Pink Paint	Decorating	6.99
Floral Wallpaper	Decorating	7.99
Magnolia Gloss Paint	Decorating	5.49
Weed Killer	Gardening	2.99
Picture Frame	Decorating	8.99
Plug Socket	Electrics	6.99
Doorbell	Electrics	15.99
Matt White Paint	Decorating	4.99
Tiles	Decorating	19.99
Grass Seed	Gardening	1.99
Lawn Mower	Gardening	129.99

Fig. 2

```
Matt Pink Paint £6.99
Floral Wallpaper £7.99
Magnolia Gloss Paint £5.49
Weed Killer £2.99
-£0.30 discount
Picture Frame £8.99
Plug Socket £6.99
Doorbell £15.99
Matt White Paint £4.99
Tiles £19.99
Grass Seed £1.99
-£0.20 discount
Lawn Mower £129.99
-£13.00 discount
----------------
TOTAL: £198.89
```

Fig. 3

Write an algorithm in pseudocode, using the array `purchases`, to:
- determine which items are given a discount
- calculate the total price to pay
- present this information on a receipt in the format shown in Fig. 3. **[6]**

1. A software company decides to release a duplicate file finder which it has named “De-Duplicator”. Duplicate files are files that are exactly the same (bit for bit identical). Space is often wasted on computers by having multiple versions of the same file. Duplicate file finders are programs that find and identify duplicate files on a hard drive so that they can be removed.

De-Duplicator creates a tree to represent directories and files on the system. It then traverses each directory and file represented in the tree. It does this using a depth-first traversal. State what order it will visit each of the **files** as shown in Fig.1 below.

--- [3]

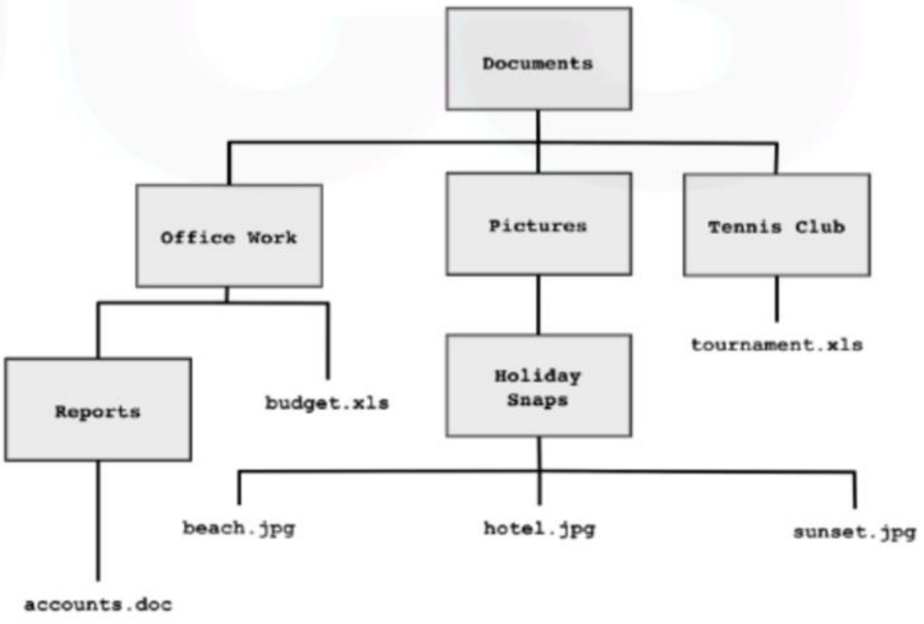
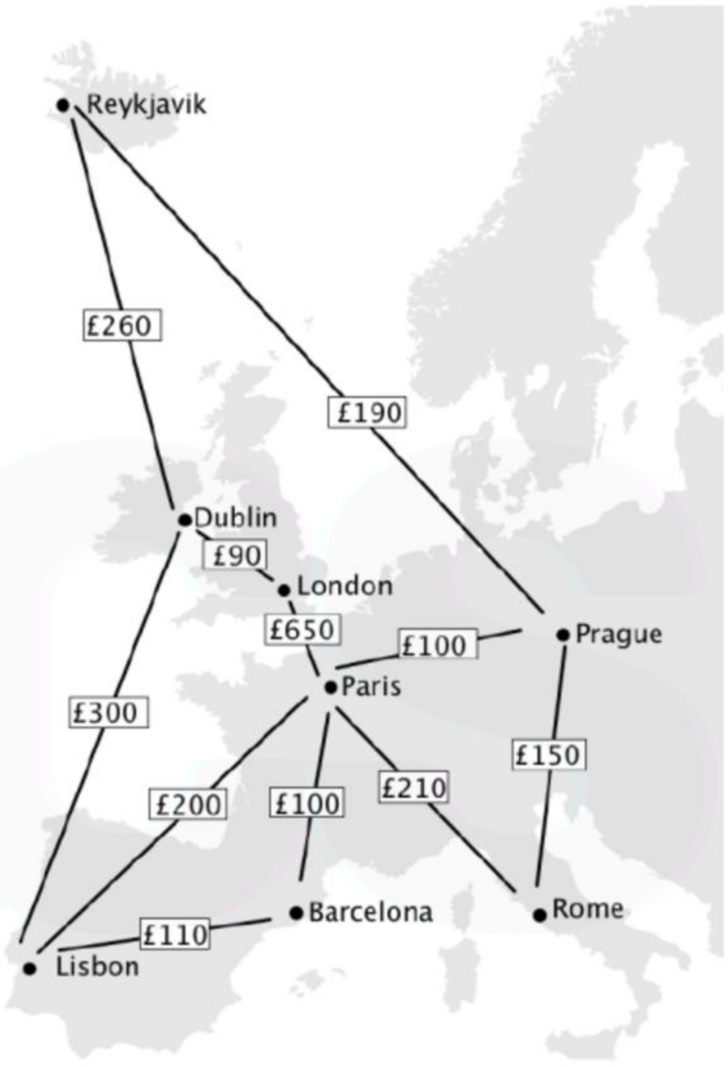


Fig.1

2(a). Atlas Airlines runs flights across cities in Europe. It stores the prices of different flights in its computer system.



State a data structure that would be suited to represent the data above.

____ [1]

(b). A function `tripCost` has been written that takes in two cities and returns the price of a direct flight between them.

e.g. `tripCost("Dublin", "London")` returns 90.

A journey is represented by an array called cities. An example of a trip from Dublin to Rome is shown below:

Dublin
London
Paris
Rome

(i) Write a program in the language or pseudocode of your choice that uses the cities array to calculate and output the cost of a given journey as a monetary value. In the case above this would be £950.

[5]

(ii) Rather than storing cities in an array you could use a linked list.

Describe a difference between an array and a linked list.

[2]

GCST

(c). Each airport has a three letter code. The airline's system stores the airports and corresponding airport codes:

Code	Name
BCN	Barcelona International
DUB	Dublin
LIS	Lisbon
LHR	London Heathrow
CDG	Paris, Charles De Gaulle
PRG	Prague
RKV	Reykjavik
FCO	Rome, Fiumicino

In a programming language or pseudocode of your choice write a program that takes in an airport code and finds and displays the airport name. You can assume a 2D array called airports has already been declared and populated with the data above. There is no need to validate the input and you can assume that the user will only enter a code that exists in the array.

[6]

3(a). A binary search tree is used to store the names of dog breeds.

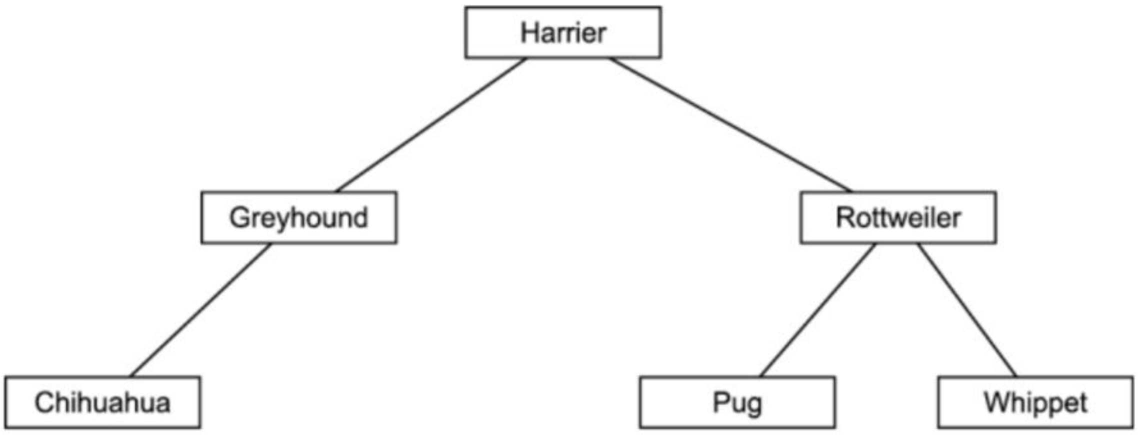


Fig. 7.1

The breeds Doberman and Dalmatian are added to the tree in that order. Add them to Fig. 7.1.

[2]

(b). Explain how you would determine if the breed Pug is in the binary search tree.

[3]

(c). Explain how you would determine if the breed Spaniel is in the binary search tree.

[3]

6. A software company is producing software that allows users with severe mobility issues to input data into a computer.

The software flashes up letters on the screen one at a time. The user sends a signal to the computer when the letter they want appears on the screen.

Rather than displaying the whole alphabet, once the first letter has been entered, the program only shows letters that could be possible according to words in its dictionary. All possible words are stored in a tree data structure.

The program is tested on a sample dictionary of four words, represented as a tree in Fig. 3:

```
BARON
BATHS
BELOW
BELTS
```

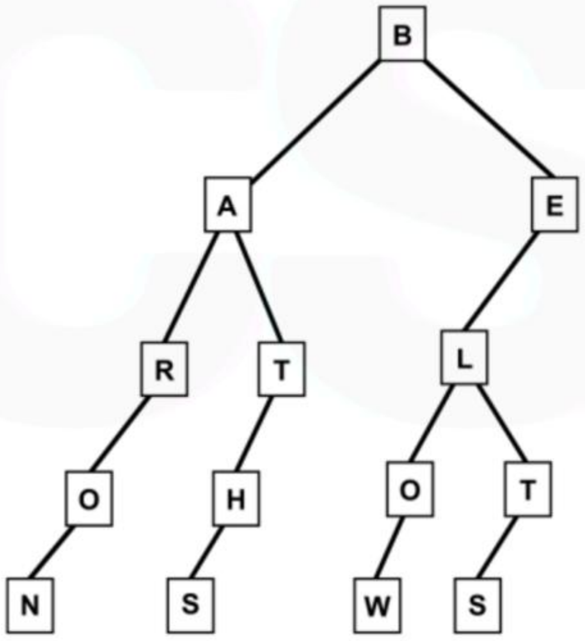


Fig. 3

(i) Annotate Fig. 3 to show how the word BELTS would be removed from the tree.

[2]

(ii) Annotate Fig. 3 to show how the words BEACH and BONE would be added to the tree.

[2]

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

THANK YOU!

GCST