

Faculty of Computing, Online Examinations 2021

STUDENT NAME	E.M. Ruchira Amantha Edirisinghe		
INDEX NUMBER (NSBM)	21487	YEAR OF STUDY AND SEMESTER	Year 1 Semester 2
MODULE NAME (As per the paper)	Algorithms & Data Structures		
MODULE CODE	CS106.3		
MODULE LECTURER	Mrs. Manoja Weerasekara	DATE SUBMITTED	8 – Sep – 2021

For office purpose only:

GRADE/MARK	
COMMENTS	

Declaration

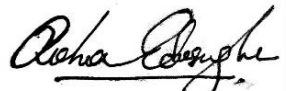
PLEASE TICK TO INDICATE THAT YOU HAVE SATISFIED THESE REQUIREMENTS

- ☐ I have carefully read the instructions provided by the Faculty
- ☐ I understand what plagiarism is and I am aware of the University's policy in this regard.
- ☐ I declare that the work hereby submitted is my own original work. Other people's work has been used (either from a printed source, Internet or any other source), has been properly acknowledged and referenced in accordance with the NSBM's requirements.
- ☐ I have not used work previously produced by another student(s) or any other person to hand in as my own.
- ☐ I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
- ☐ I hereby certify that the individual detail information given (name, index number and module details) in the cover page are thoroughly checked and are true and accurate.

I hereby certify that the statements I have attested to above have been made in good faith and are true and correct. I also certify that this is my own work and I have not plagiarized the work of others and not participated in collusion.

Date: **8 – Sep – 2021**

**E- Signature:



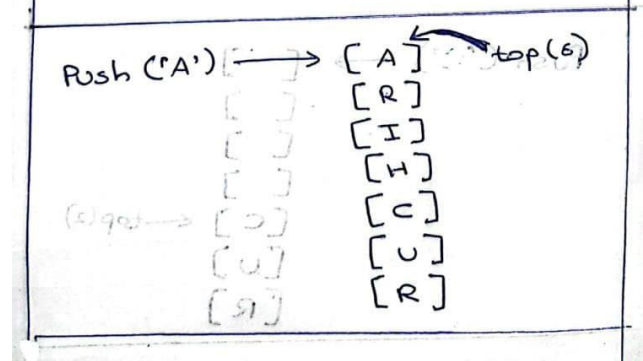
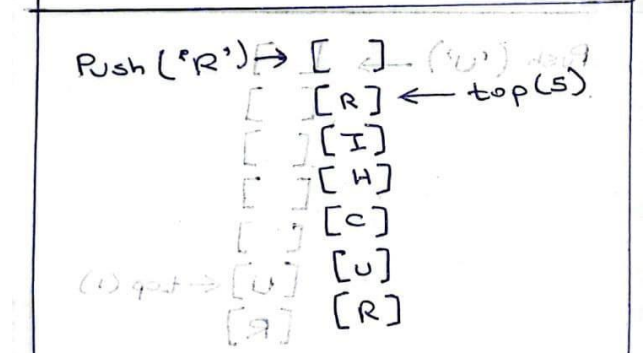
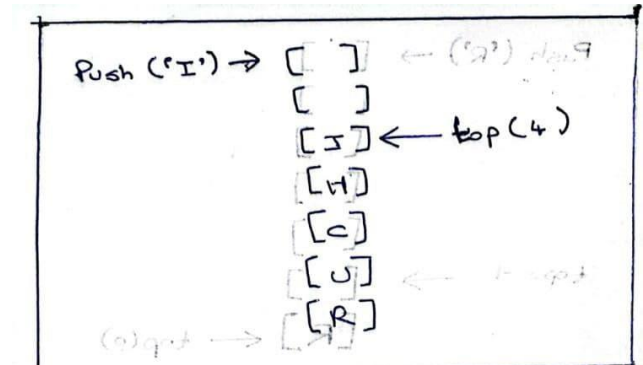
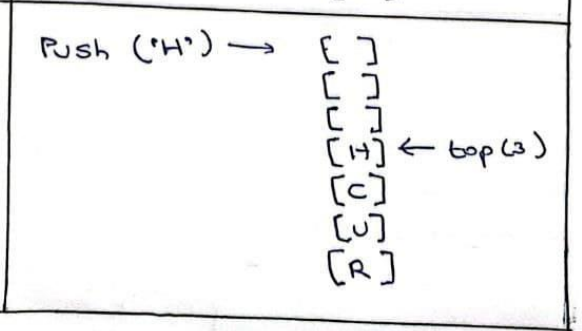
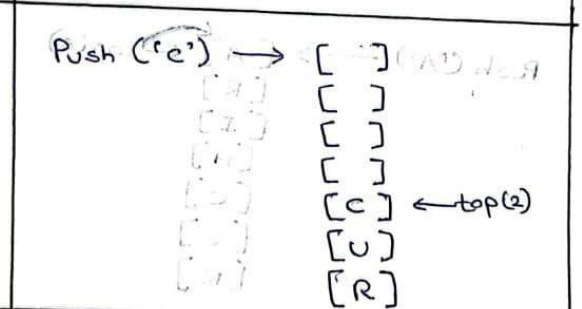
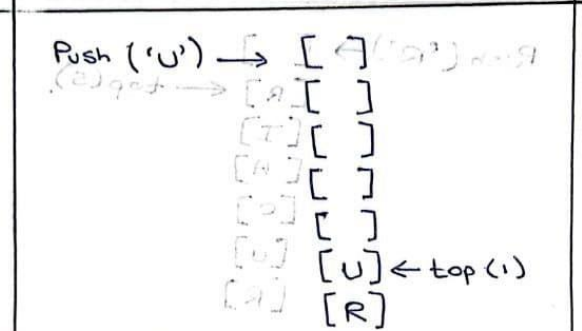
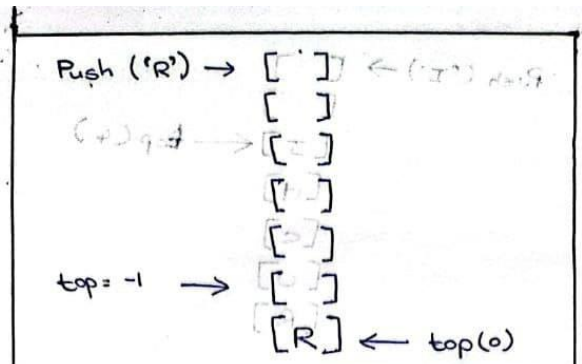
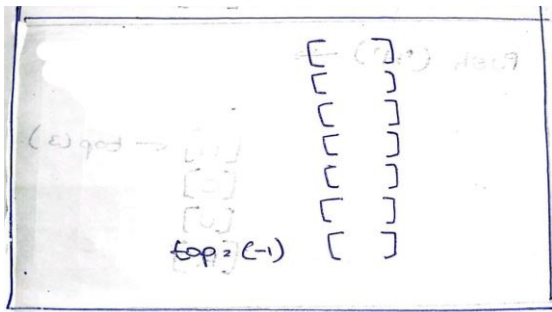
**Please attach a photo/image of your signature in the space provided.

Question 1

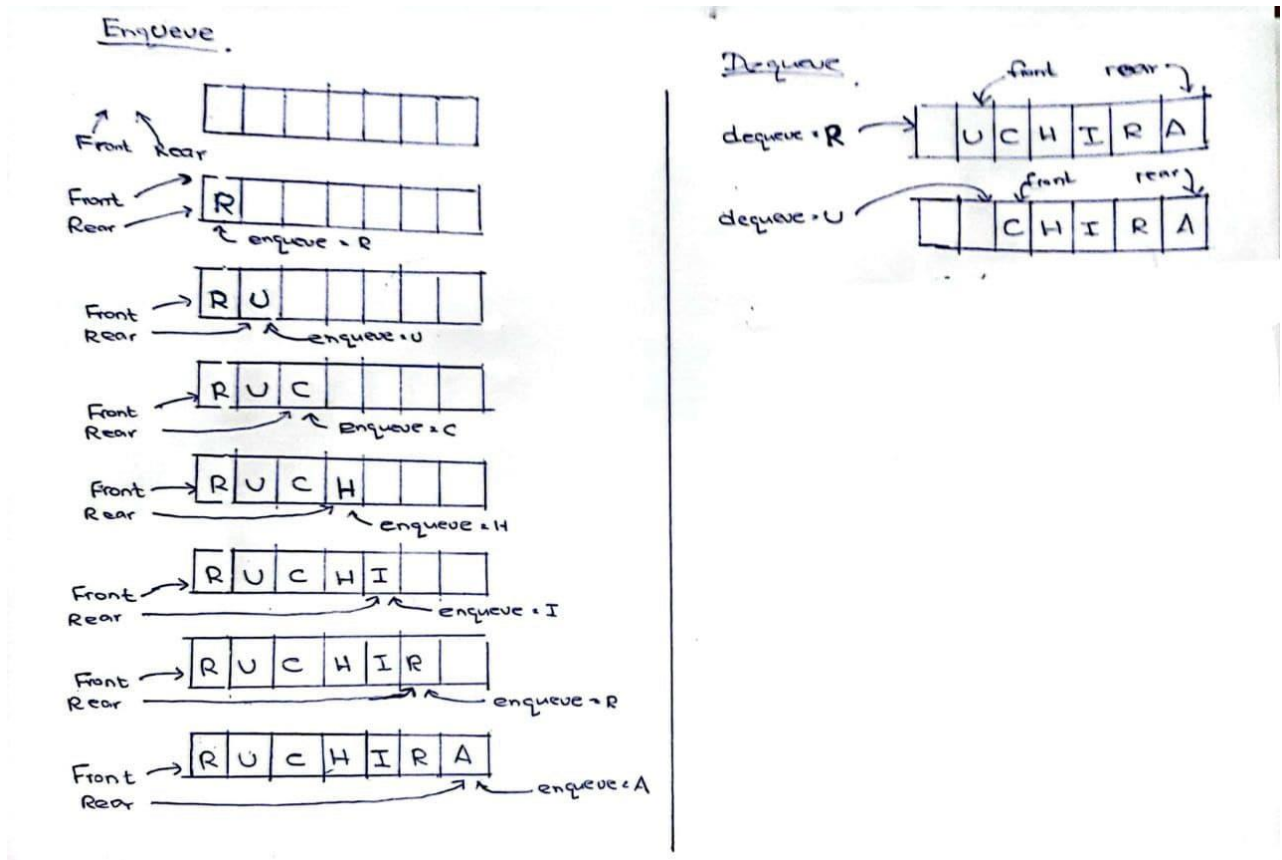
1.

Stack	Queue	Circular Queue
Objects are inserted and removed at the same end.	Objects are inserted and removed from different ends	Can enter new items and remove items from any position.
More Efficient than queue.	Efficient, but not like Stack.	Efficient. Circular queue is more efficient than linear queue.
Uses Last in First Out Method. (LIFO)	Uses First in First Out Method. (FIFO)	Uses First in First Out Method. (FIFO)
Only one pointer is used, and it is pointed to the top of the stack.	Two different pointers are used for front and back.	Two pointers are used, head pointer points to the front of the queue and tail pointer points to the end of the queue.
One end of the list, called as "the top", is used for insertion and deletion.	Allows only for insertion and deletion from the back and front respectively.	Not limited to a specific location, and this can be done at any position.

2.



3.



4.

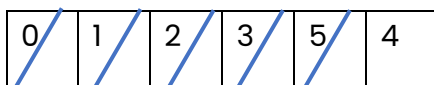
Enqueue () Function

void enqueue (int item)

```
{
    st.rear++;
    st.q[st.rear]=item;
    print ("%d - Added Successfully",item)
}
```

5. In BFS, we use the queue data structure, to implement it.

BFS → 0, 1, 2, 3, 5, 4



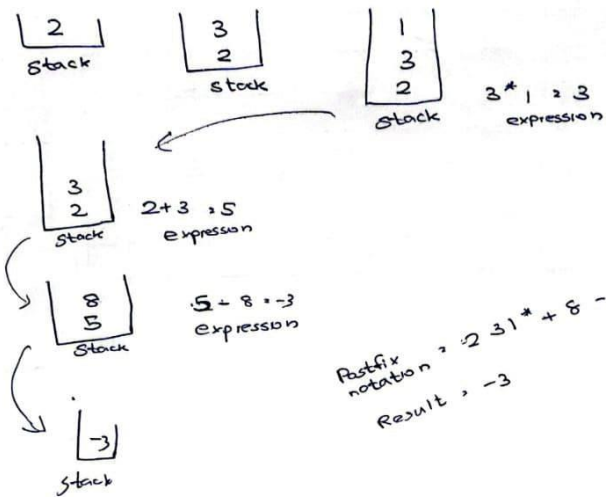
In DFS, we use the stack data structure to implement it.

DFS \rightarrow 0, 1, 2, 4, 3, 5

5
3
4
2
1
0

6.

(a)



(b)

6 2 3 + - 3 8 2 / + * 2 ^ 2 +

Stack: 3, 2, 6

$$2 + 3 = 5$$

Stack: 5, 6

$$6 - 5 = 1$$

Stack: 2, 8, 3, 1

$$8 / 2 = 4$$

Stack: 4, 3, 1

$$5 + 4 = 9$$

Stack: 7

$$7 * 1 = 7$$

Stack: 2, 7

$$7^2 = 49$$

Stack: 2, 49

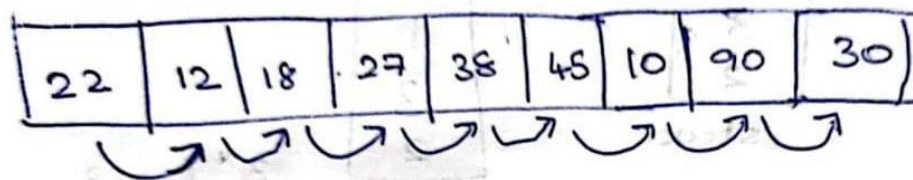
$$49 + 2 = 51$$

Result = 51
Stack is the suitable structure

Question 2

1. When linear search and binary search are compared with each other, the linear search has a less complexity and less efficiency than the binary search. Finding the element in the first index is the best case of the binary search and finding the element in the middle index is the best case of binary search. The time complexity of the linear search is $O(n)$ and the time complexity of the Binary search is $O(\log_2 n)$.

2.



Search key = 30

sk = search key

arr[0] = sk;

arr[1] = sk;

arr[2] = sk;

arr[3] = sk;

arr[4] = sk;

arr[5] = sk;

arr[6] = sk;

arr[7] = sk;

arr[8] = sk; "Item found"

3.

Case 1 ,

0	1	2	3	4	5	6	7	8
10	12	18	22	27	30	38	45	90

$Sk = 30$

$$\text{Mid value} = \frac{0+8}{2}$$

♦ Index = 4

Value of index 4 = 27

$Sk > 27$ ∴ we have to consider the right side
index 4

Case 2 ,

4	5	6	7	8
27	30	38	45	90

$$\text{Mid value} = \frac{4+8}{2}$$

index = 6

Value of index 6 = 38

$Sk < 38$ ∴ we must consider the left side of
the array.

Case 3 ,

4	5	6
27	30	38

$$\text{Mid value} = \frac{4+6}{2}$$

index = 5

Value of index 5 = 30

$Sk = 30$

value found

4. When there are 1024 values in the array, finding the value of the 1st index (0th index) is the best-case scenario of linear search and finding the value in the last index which is 1023rd index is the worst-case scenario of it.
- But when there are 1024 values in the array, finding the search value in the first middle value which is 511th index is the best-case scenario in binary search.

Question 3

1.

Pass 0

22	12	18	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

12 22 18 27 38 45 10 90 30

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

12	18	22	27	38	10	45	90	30
----	----	----	----	----	----	----	----	----

12	18	22	27	38	10	45	90	30
----	----	----	----	----	----	----	----	----

12	18	22	27	38	10	45	30	90
----	----	----	----	----	----	----	----	----

Pass 1

12	18	22	27	38	10	45	30	90
----	----	----	----	----	----	----	----	----

12	18	22	27	10	38	30	45	90
----	----	----	----	----	----	----	----	----

12	18	22	27	10	38	30	45	90
----	----	----	----	----	----	----	----	----

Pass 2

12	18	22	27	10	38	30	45	90
12	18	22	10	27	38	30	45	90
12	18	22	10	27	30	38	45	90

Pass 3

12	18	22	10	27	30	38	45	90
12	18	10	22	27	30	38	45	90

Pass 4

12	18	10	22	27	30	38	45	90
12	10	18	22	27	30	38	45	90

Pass 5

12	10	18	22	27	30	38	45	90
10	12	18	22	27	30	38	45	90

Pass 6

10	12	18	22	27	30	38	45	90
----	----	----	----	----	----	----	----	----

Pass 7

10	12	18	22	27	30	32	45	90
----	----	----	----	----	----	----	----	----

The last element processed.
It's in its final position.
Then sorting is completed.

10	12	18	22	27	30	32	45	90
----	----	----	----	----	----	----	----	----

Date: / /

No:

sorted

Iteration
1;

22		18	27	38	45	10	90	30
----	--	----	----	----	----	----	----	----

tem = 12

Unsorted

compare 22 & 12

12	22	18	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

Swap = 12

sorted

Iteration
2;

12	22		27	38	45	10	90	30
----	----	--	----	----	----	----	----	----

tem = 18

Unsorted

compare 18 & 22

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

Swap = 18

sorted

Iteration
3;

12	18	22		38	45	10	90	30
----	----	----	--	----	----	----	----	----

tem = 27

Unsorted

compare 27 & 22

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

no swap

sorted

Iteration
4;

12	18	22	27		45	10	90	30
----	----	----	----	--	----	----	----	----

tem = 38

Unsorted

compare 27 & 38

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

no swap

sorted

Iteration
5;

12	18	22	27	38		10	90	30
----	----	----	----	----	--	----	----	----

tem = 45

Unsorted

compare 45 & 38

12	18	22	27	38	45	10	90	30
----	----	----	----	----	----	----	----	----

no swap

sorted

Iteration
6;

12	18	22	27	38	45		90	30
----	----	----	----	----	----	--	----	----

tem = 10

Unsorted

10	12	18	22	27	38	45	90	30
----	----	----	----	----	----	----	----	----

Swap = 10

sorted

Iteration
7;

10	12	18	22	27	38	45		30
----	----	----	----	----	----	----	--	----

tem = 90

Unsorted

compare 90 & 45

10	12	18	22	27	38	45	90	30
----	----	----	----	----	----	----	----	----

no swap

sorted

Iteration
8;

10	12	18	22	27	38	45	90	
----	----	----	----	----	----	----	----	--

tem = 30

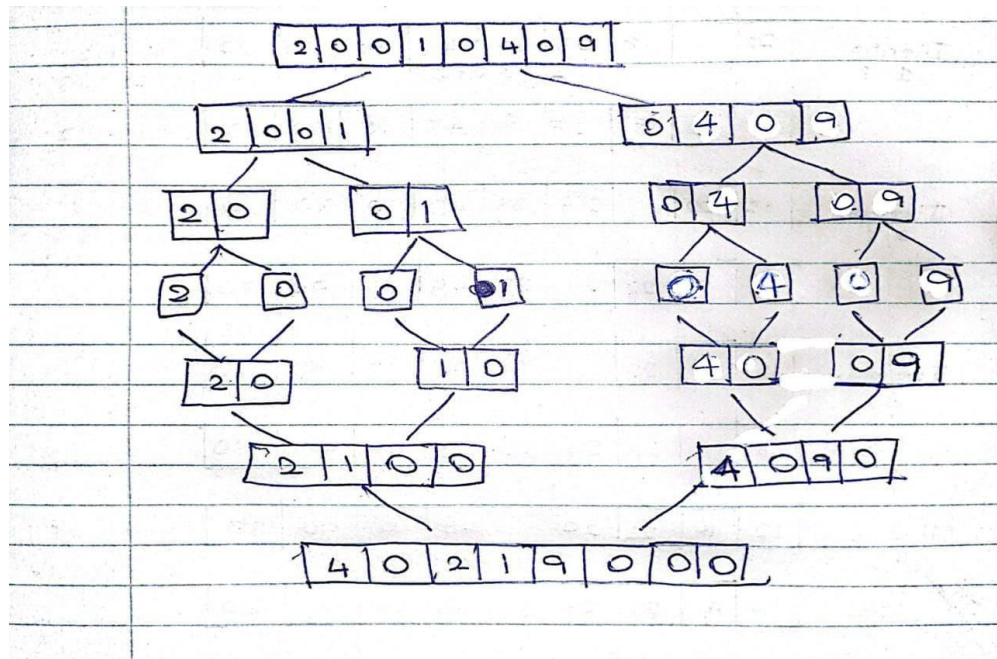
compare 30 & 90

10	12	18	22	27	30	38	45	90
----	----	----	----	----	----	----	----	----

Swap 30

RICHARD

3.



4.

- Insertion can be very useful when the number of elements inside the array is too small.
- Worst Case scenario in the insertion sort algorithms happens in the order of the input list in descending order.

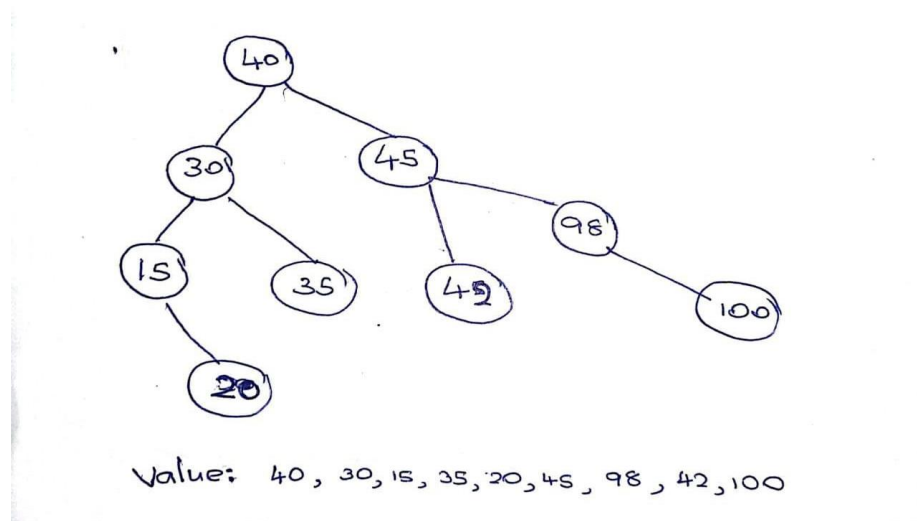
```

for ( i=0; i<n; i++)
{
    temp = A[i];
    j = i-1;
    while(j>=0 && A[j]>temp)
    {
        A[j+1] =A[j];
        j--;
    }
    A[j+1] =temp;
}

```


Question 4

1.



2. Pre-order – 40, 30, 15, 20, 35, 45, 42, 98, 100

Past-order – 20, 15, 35, 30, 42, 100, 98, 45, 40

In-order – 15, 20, 30, 35, 40, 42, 45, 98, 100

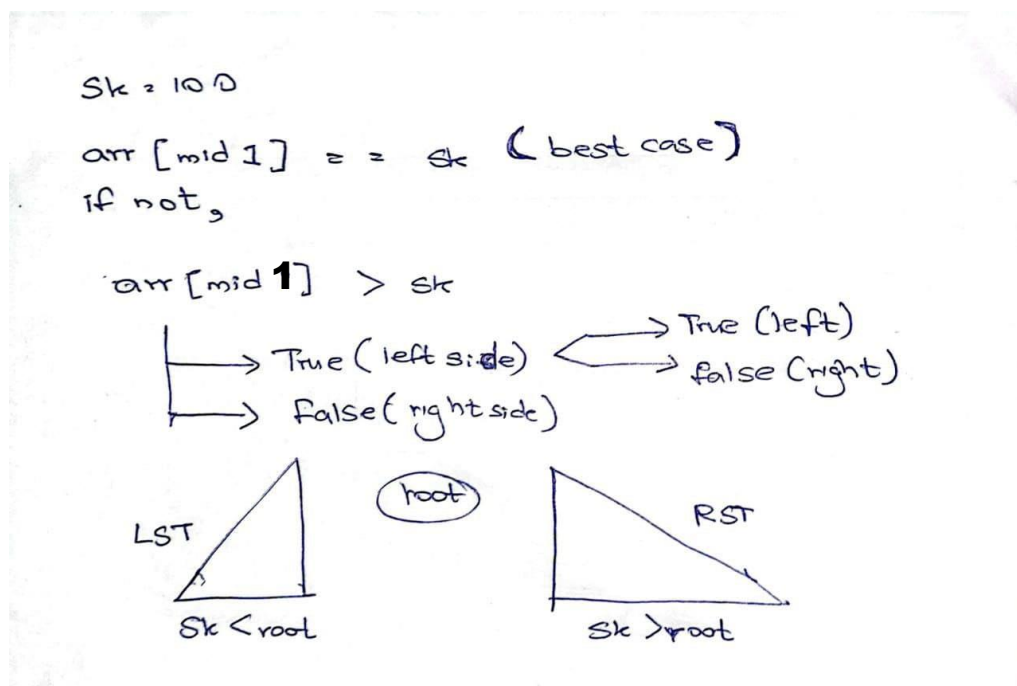
3. Depth of the node 98 = 2

4. $20 + 35 + 42 + 100 = \underline{197}$

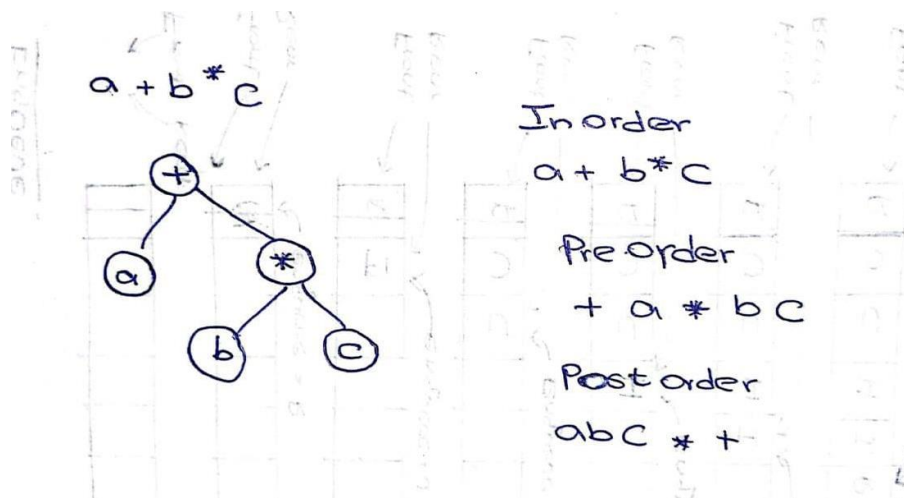
5. No it cannot be considered as Perfect Binary Tree (Too many Nodes)

Height $\rightarrow \underline{5}$

6.



7.



8.

