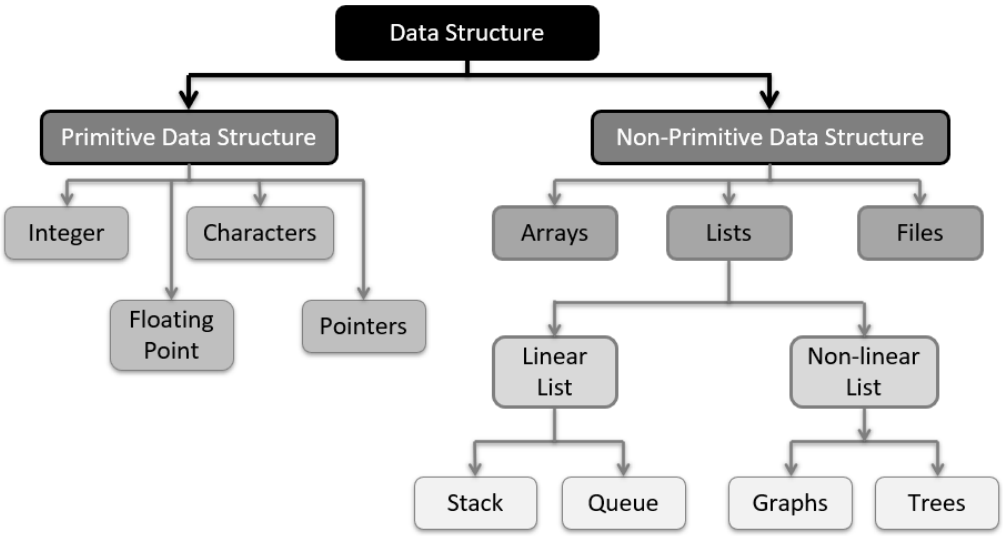
Answer Key-

Question-1

1. Draw the classification diagram of Data Structure.

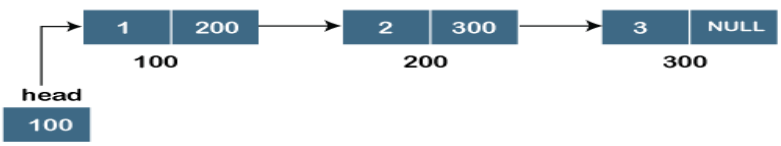


2. ADT stands for _____
   Abstract Data Type

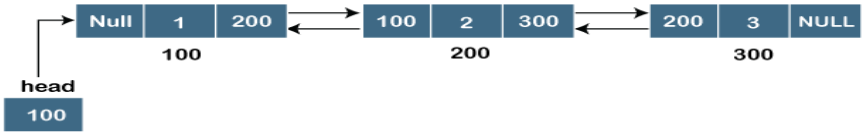
3. List any four operations of Data Structure.
   a. Create
   b. Destroy
   c. Selection
   d. Updation
   e. Searching
   f. Sorting
   g. Merging
   h. Splitting
   i. Traversal

4. Stack follows FILO manner. True or False?
   True

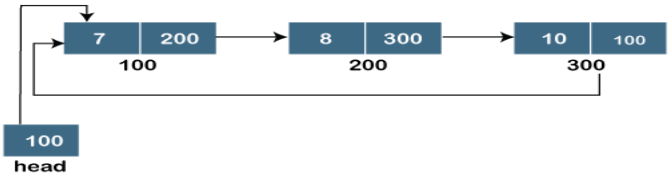
5. Give pictorial representation of all types of Linked List.
   Singly Linked List



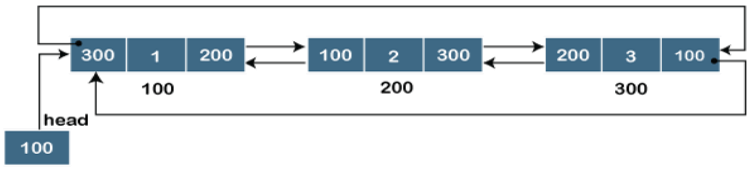
Doubly Linked List



Circular Singly Linked List



Circular Doubly Linked List



6. Write a formula to find MID in binary search algorithm.
MID = (high + low)/2

Question-2

(a) Write algorithms for Enqueue() and Dequeue() for Simple queue.

Enqueue- 4 marks

1. [check for queue overflow]

   if REAR == MAX -1

   then Write("queue overflow")

   Return

2. [update pointers]

   if FRONT == -1 and REAR == -1

   then SET FRONT = REAR = 0 else SET REAR = REAR + 1

3. [insert element] Q[REAR] = X
4. [finished]

   Return

Dequeue- 4 marks

DEQUEUE(X)

1. [check for underflow of the queue]

   if FRONT == -1OR FRONT > REAR

   thenWrite("queue underflow")

   Return

2. [get front element of the queue] Value =Q[FRONT]
3. [update pointers] FRONT = FRONT + 1
4. [finished]

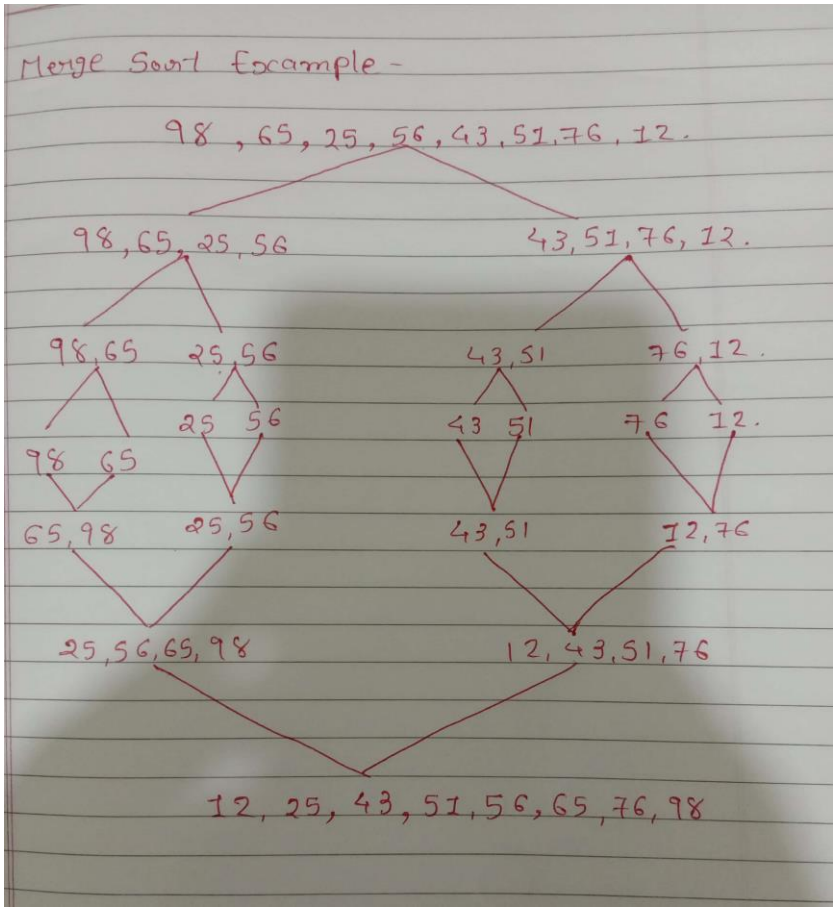
   ReturnValue

(b) Write an algorithm for Merge Sort and sort the given data using the same- [6]
98, 65, 25, 56, 43, 51, 76, 12

Algorithm – 3 marks

```
MergeSort(int arr[], int size){
        int* tmp=new int[size];
        MSort(arr, tmp, 0, size-1);
        delete [] tmp;
}
MSort(int arr[], int tmp[], int start, int end){
        if(start<end){
                int mid=(start+end)/2;
                MSort(arr, tmp, start, mid);
                MSort(arr, tmp, mid+1, end);
                Merge(arr, tmp, start, mid+1, end);
        }
}
```

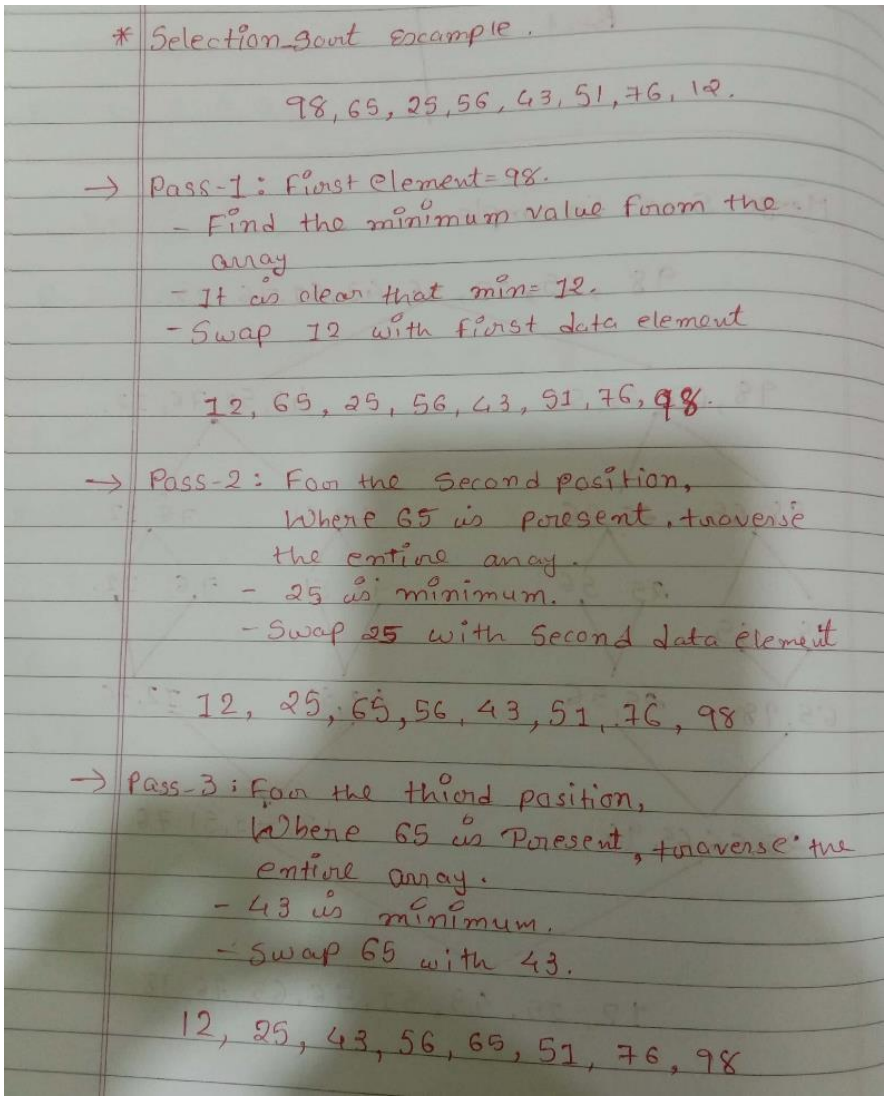
Example – 3 marks



OR

(b)Sort the given data using the selection sort- 8 marks

98, 65, 25, 56, 43, 51, 76, 12



→ Pass-4: For the fourth position,
- Where 56 is present, traverse the entire array.
- 51 is minimum.
- Swap 56 with 51.

12, 25, 43, 51, 65, 56, 76, 98.

→ Pass-5: For the Fifth position,
- Where 65 is present, traverse the entire array.
- 56 is minimum.
- Swap 65 with 56.

12, 25, 43, 51, 56, 65, 76, 98.

→ Pass-6: For the Sixth position,
- Where 65 is present, traverse the entire array.
- 65 itself is the smallest
- 65 will be as it is.

12, 25, 43, 51, 56, 65, 76, 98.

→ Pass-7: For the Seventh position,
- Where 76 is present, traverse the entire array.
- 76 itself is the Smallest
- 76 will be as it is

12, 25, 43, 51, 56, 65, 76, 98

Pass-8: For the Eighth position,
- Find the minimum from the entire array.
- 98 will be as it is.

12, 25, 43, 51, 56, 65, 76, 98

Sorted array.

Question-3

(a) Write down algorithms for PUSH() the data into stack and POP() the data from stack with example.

PUSH – 4 marks

ALGORITHM: PUSH(S,X)

1. [check for stack overflow]

if S->TOP = MAX

thenWrite("stack overflow")

Return

2. [increment top]

S->TOP = S->TOP + 1

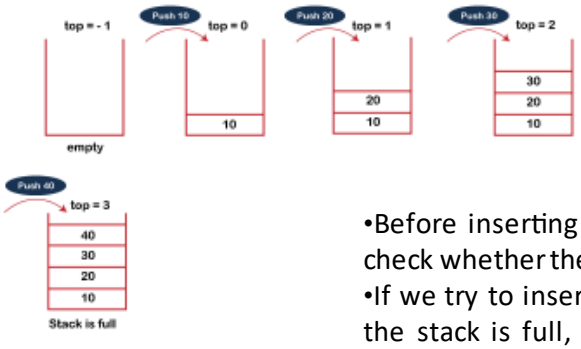
3. [insert element] S[S->TOP] = X
4. [finished]

Return

Push Operation

Marwadi University

Push an item onto the top of the stack (insert an item)



- Before inserting an element in a stack, we check whether the stack is full.
- If we try to insert the element in a stack, and the stack is full, then the ***overflow*** condition occurs.

POP – 4 marks

ALGORITHM: POP(S)

1. [check for underflow of the stack] if S->TOP = NULL

then Write("stack underflow")

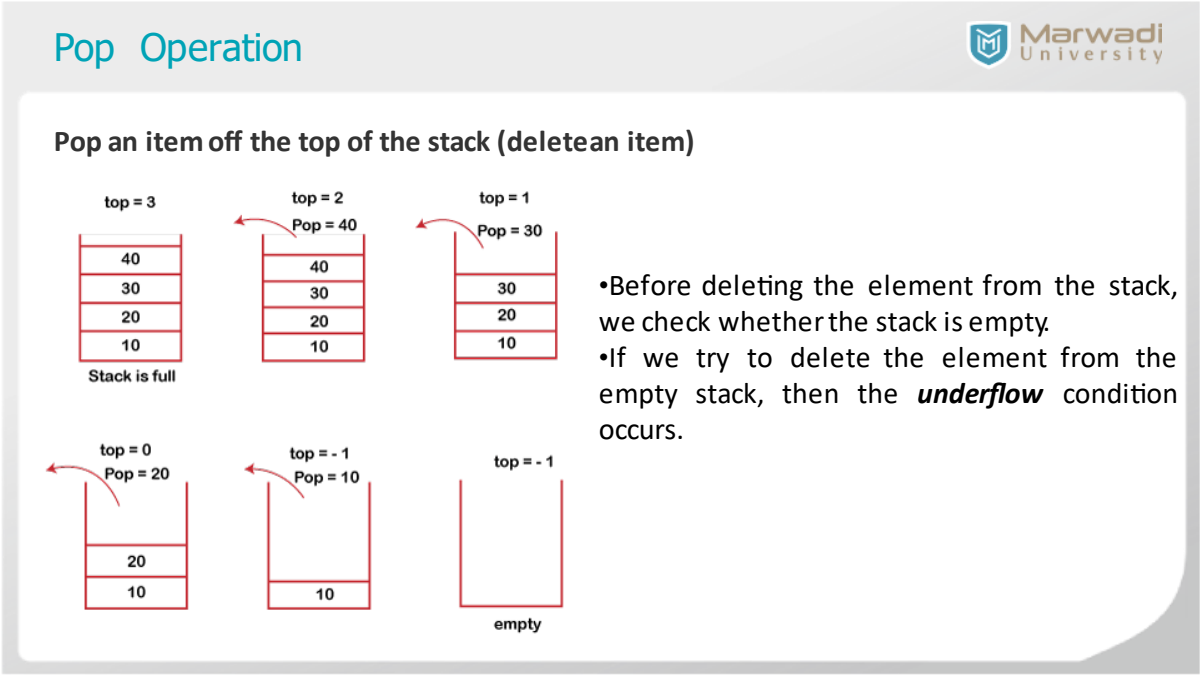
Return

2. [get former top element of the stack]

Value = S[S->TOP]

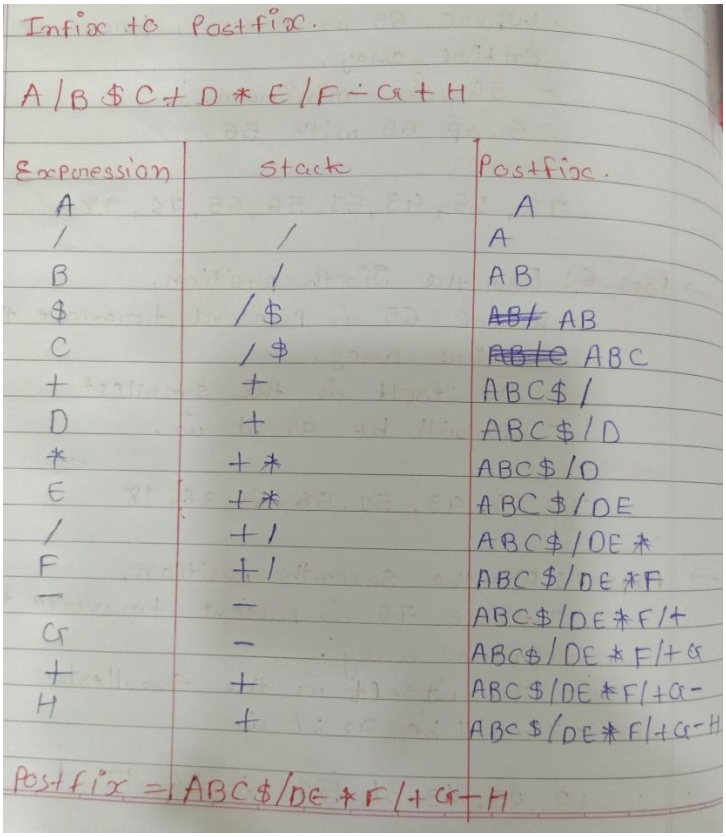
3. [decrement pointer] S->TOP = S->TOP - 1
4. [finished]

Return Value



(b) Convert the given Infix expression into Postfix using Stack

A / B $ C + D * E / F – G + H. - 4 marks



Infix to Postfix.

A / B $ C + D * E / F − G + H

Expression	stack	Postfix.
A		A
/	/	A
B	/	AB
$	/ $	~~AB/~~ AB
C	/ $	~~AB$C~~ ABC
+	+	ABC$/
D	+	ABC$/D
*	+ *	ABC$/D
E	+ *	ABC$/DE
/	+ /	ABC$/DE*
F	+ /	ABC$/DE*F
−	−	ABC$/DE*F/+
G	−	ABC$/DE*F/+G
+	+	ABC$/DE*F/+G-
H	+	ABC$/DE*F/+G-H

Postfix = ABC$/DE*F/+G-H

OR

(a) Write algorithms for Insertion at first and Insertion at end for Doubly Linked List. Also write down applications of Linked List.

Insertion at first – 3 marks

Step 1: IF AVAIL = NULL, then Write OVERFLOW

Go To Step 9 [END OF IF]

Step 2: SET New_Node= AVAIL

Step 3: SET AVAIL = AVAIL -> NEXT

Step 4: SET New_Node-> DATA = VAL

Step 5: SET New_Node-> PREV = NULL

Step 6: SET New_Node-> NEXT = START

Step 7: SET START -> PREV = New_Node

Step 8: SET START = New_Node

Step 9: Exit

Insertion at end – 3 marks

Step 1: IF AVAIL = NULL, then

Write OVERFLOW Go To Step 11

[END OF IF]

Step 2: SET New_Node= AVAIL

Step 3: SET AVAIL = AVAIL -> NEXT

Step 4: SET New_Node-> DATA = VAL

Step 5: SET New_Node-> NEXT = NULL

Step 6: SET PTR = START

Step 7: Repeat Step 8 while PTR->NEXT != NULL

Step 8: SET PTR = PTR -> NEXT [END OF LOOP]

Step 9: SET PTR -> NEXT = New_Node

Step 10: New_Node-> PREV = PTR

Step 11: Exit

Applications of LinkedList – 2 marks

- In web browsers, you might have seen that we can always access the previous and next URL using the back and forward button.
- Access to previous and next URL searched is possible because they are linked using a linked list.
- The songs in the Music Player are linked to the next and the previous song. We can play songs either from the starting or the end of the list.
- In an Image Viewer, the next and the previous images are linked; hence they can be accessed by the previous and the next button.

(b) Write down four applications of stack and four applications of queue.

Applications of stack – 2 marks

- Stacks can be used for Conversion from one form of expression to another. -POLISH and REVERSE POLISH NOTATIONS
- Stacks can be used for expression evaluation.
- Recursion-Tower of Hanoi
- Stacks can be used to check parenthesis matching in an expression.
- Stacks can be used for Memory Management.

2. Stack data structures are used in backtracking problems.

Applications of Queue – 2 marks

- Queues are widely used as waiting lists for a single shared resource like printer, disk, CPU.
- Queues are used to transfer data asynchronously (data not necessarily received at same rate as sent) between two processes (IO buffers), e.g., pipes, file IO, sockets.
- Queues are used as buffers on MP3 players and portable CD players, iPod playlist.
- Queues are used in Playlist for jukebox to add songs to the end, play from the front of the list.
- Queues are used in operating system for handling interrupts. When programming a real-time system that can be interrupted, for example, by a mouse click, it is necessary to
- process the interrupts immediately, before proceeding with the current job. If the interrupts have to be handled in the order of arrival, then a FIFO queue is the appropriate data structure