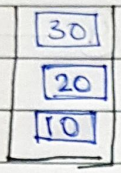
Page No.
Date :

STACKS

- STACK is a linear data structure
- Works on LAST IN FIRST OUT



UNDERFLOW: When we pop() from an empty stack.

OVERFLOW: When we push() to an already filled stack

APPLICATIONS OF STACK

- Function Calls (Stack)
- Checking for balanced parenthesis
- Reversing items
- Infix to Prefix / Postfix
- Evaluation of Postfix / prefix
- Stock span problems
- ⟹ UNDO / REDO or FORWARD / BACKWARD

Page No.
Date :

STACK IMPLEMENTATION USING ARRAYS

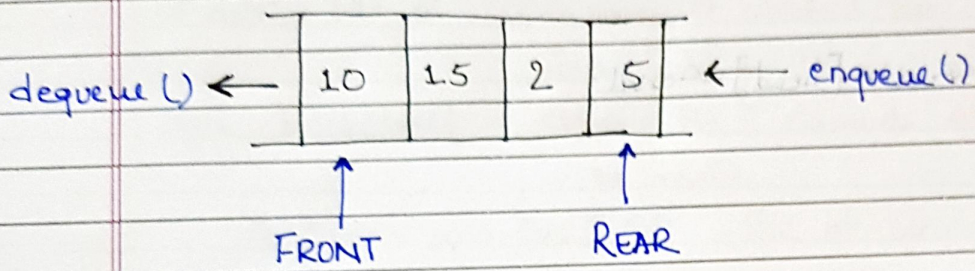
```
#include <bits/stdc++.h>
using namespace std;
#define MX 5   //An array of constant size has been defined since doubling the size of array and copying the elements after OVERFLOW is expensive.
class stackClass{   Use of LINKED LIST>>>
    int * arr= new int [5];
    int val;
    int head =-1;

  public:
    void push (int val) {
      if (head == MX-1) {
          cout<<"Stack Overflow! \n";
          return;
      }
     head += 1;
     arr [head ]= val;
}

 void pop() {
    if (head == -1){
       cout<<" stack Underflow! \n";
       return;
    }
     head -= 1;
 }
```

```
void top(){
    if(head == -1){
        cout<<"Stack empty. \n";
        return;
    }
    cout<<arr[head]<<endl;
  }
};

int main(){

    stack Class  s1,s2;
    s1.top();
    s1.push(1);
    s1.push(2);
    s1.push(3);
    s1.pop();
    s1.top();
    s2.pop();
}
```

Page No.
Date :

QUEUE IMPLEMENTATION USING ARRAYS

FIRST IN FIRST OUT



Operations:

→ Enqueue (x);
→ Dequeue ();
→ Getfront ();
→ Get Rear ();
→ Size ();
→ isEmpty ();

APPLICATIONS:

→ Single resource and Multiple consumers
→ Synchronization between slow and fast devices
→ In OS (Semaphores, FCFS, Spooling, buffers)
→ In Computer Networks (Routers/switches and Mail queues)

VARIATIONS:

- Deque ⇒ Doubly-ended queue
- Priority Queue ⇒ Implemented using HEAP
- Doubly ended priority queue