STACK

1. STACK USING LIST

```
stack = []
stack.append(10)
stack.append(20)
stack.append(30)
print(stack)
print(stack.pop())

top=stack[-1]
print(top)
size=len(stack)
print(size)
```

2. STACK USING DEQUE

```
from _collections import deque
stack =deque()
stack.append(10)
stack.append(20)
stack.append(30)
print(stack)
print(stack.pop())

top=stack[-1]
print(top)
size=len(stack)
print(size)
```

3. STACK USING ARRAY

```
#include <bits/stdc++.h>
using namespace std;

struct MyStack{
    int *arr;
    int cap;
    int top;
    MyStack(int c){
        cap=c;
        arr=new int [cap];
        top=-1;
    }

    void push(int x){
        if(top==cap-1){cout<<"Stack is full"<<endl;return;}
        top++;
        arr[top]=x;
    }

    int pop(){
        if(top==-1){cout<<"Stack is Empty"<<endl;return INT_MIN;}
        int res=arr[top];
        top--;
        return res;
    }

    int peek(){
        if(top==-1){cout<<"Stack is Empty"<<endl;return INT_MIN;}
        return arr[top];
    }

    int size(){
        return (top+1);
```

```
    }

    bool isEmpty(){
        return top==-1;
    }
};

int main()
{
    MyStack s(5);
    s.push(5);
    s.push(10);
    s.push(20);
    cout<<s.pop()<<endl;
    cout<<s.size()<<endl;
    cout<<s.peek()<<endl;
    cout<<s.isEmpty()<<endl;

    return 0;
}
```

4. STACK USING VECTOR

```
#include <bits/stdc++.h>
using namespace std;

struct MyStack{
    vector<int> v;

    void push(int x){
        v.push_back(x);
    }

    int pop(){
```

```
        int res=v.back();
        v.pop_back();
        return res;
    }

    int peek(){
        return v.back();
    }

    int size(){
        return v.size();
    }

    bool isEmpty(){
        return v.empty();
    }
};

int main()
{
    MyStack s;
    s.push(5);
    s.push(10);
    s.push(20);
    cout<<s.pop()<<endl;
    cout<<s.size()<<endl;
    cout<<s.peek()<<endl;
    cout<<s.isEmpty()<<endl;

    return 0;
}
```

5. LINKED LIST IMPLEMENTATION OF STACK

```
#include <bits/stdc++.h>
using namespace std;

struct Node{
    int data;
    Node *next;
    Node(int x){
        data=x;
        next=NULL;
    }
};

struct MyStack{
    Node *head;
    int sz;
    MyStack(){
        head=NULL;
        sz=0;
    }

    void push(int x){
        Node *temp=new Node(x);
        temp->next=head;
        head=temp;
        sz++;
    }

    int pop(){
        if(head==NULL){cout<<"Stack is Empty"<<endl;return
INT_MAX;}
        int res=head->data;
        Node *temp=head;
        head=head->next;
        delete(temp);
        sz--;
```

```
        return res;
    }

    int peek(){
        if(head==NULL){cout<<"Stack is Empty"<<endl;return
INT_MAX;}
        return head->data;
    }

    int size(){
        return sz;
    }

    bool isEmpty(){
        return head==NULL;
    }
};

int main()
{
    MyStack s;
    s.push(5);
    s.push(10);
    s.push(20);
    cout<<s.pop()<<endl;
    cout<<s.size()<<endl;
    cout<<s.peek()<<endl;
    cout<<s.isEmpty()<<endl;

    return 0;
}
```

6. STACK IN C++ STL

```
#include <iostream>
#include <stack>
using namespace std;

int main()
{
    stack<int> s;
    s.push(10);
    s.push(20);
    s.push(30);
    cout<<s.size()<<endl;
    cout<<s.top()<<endl;
    s.pop();
    cout<<s.top()<<endl;
    s.push(5);
    cout<<s.top()<<endl;

    while(s.empty()==false){
        cout<<s.top()<<endl;
        s.pop();
    }
    return 0;
}
```

7. BALANCED PARENTHESIS

```
#include <bits/stdc++.h>
using namespace std;

bool matching(char a,char b){
    return (( a=='(' && b==')' )||( a=='[' && b==']' )||( a=='{' && b=='}' ));
}

bool isBalanced(string str){
```

```
    stack<char> s;

    for (int i = 0; i < str.length(); i++)
    {
        if (str[i] == '(' || str[i] == '[' || str[i] == '{')
        {
            s.push(str[i]);
        }
        else{
        if (s.empty()==true)
            return false;
        else if(matching(s.top(),str[i])==false)
            return false;
        else
            s.pop();
        }
    }
    return (s.empty()==true);
}

int main()
{
    string str = "{()}[]";

    if (isBalanced(str))
        cout << "Balanced";
    else
        cout << "Not Balanced";

    return 0;
}
```

8. 2 STACKS IN AN ARRAY

```
#include <bits/stdc++.h>
using namespace std;

struct TwoStacks {
    int* arr;
    int cap;
    int top1, top2;

    TwoStacks(int n)
    {
        cap = n;
        arr = new int[n];
        top1 = -1;
        top2 = cap;
    }

    void push1(int x)
    {
        if (top1 < top2 - 1) {
            top1++;
            arr[top1] = x;
        }
        else {
            cout << "Stack Overflow";
            exit(1);
        }
    }

    void push2(int x)
    {
        if (top1 < top2 - 1) {
            top2--;
```

```
            arr[top2] = x;
        }
        else {
            cout << "Stack Overflow";
            exit(1);
        }
    }

    int pop1()
    {
        if (top1 >= 0) {
            int x = arr[top1];
            top1--;
            return x;
        }
        else {
            cout << "Stack UnderFlow";
            exit(1);
        }
    }

    int pop2()
    {
        if (top2 < cap) {
            int x = arr[top2];
            top2++;
            return x;
        }
        else {
            cout << "Stack UnderFlow";
            exit(1);
        }
    }
};
```

```
int main()
{
    TwoStacks ts(5);
    ts.push1(5);
    ts.push2(10);
    ts.push2(15);
    ts.push1(11);
    ts.push2(7);
    cout << "Popped element from stack1 is "<<ts.pop1();
    ts.push2(40);
    cout << "\nPopped element from stack2 is "<< ts.pop2();
    return 0;
}
```

9. K STACKS IN AN ARRAY

```
#include <bits/stdc++.h>
using namespace std;

struct kStacks
{
    int *arr;
    int *top;
    int *next;
    int cap, k;
    int freeTop;

    kStacks(int k1, int n){
        k = k1; cap = n;
        arr = new int[cap];
        top = new int[k];
        next = new int[cap];

        for (int i = 0; i < k; i++)
```

```
        top[i] = -1;

    freeTop = 0;
    for (int i=0; i<cap-1; i++)
        next[i] = i+1;
    next[cap-1] = -1;
}

bool isFull()   {  return (freeTop == -1);  }

bool isEmpty(int sn)  {  return (top[sn] == -1); }

void push(int x, int sn)
{
if (isFull())
{
    cout << "\nStack Overflow\n";
    return;
}

int i = freeTop;
freeTop = next[i];
next[i] = top[sn];
top[sn] = i;
arr[i] = x;
}

int pop(int sn)
{
if (isEmpty(sn))
{
     cout << "\nStack Underflow\n";
     return INT_MAX;
}
```

```
    int i = top[sn];
    top[sn] = next[i];
    next[i] = freeTop;
    freeTop = i;
    return arr[i];
    }

};

int main()
{
    int k = 3, n = 10;
    kStacks ks(k, n);

    ks.push(15, 2);
    ks.push(45, 2);

    ks.push(17, 1);
    ks.push(49, 1);
    ks.push(39, 1);

    ks.push(11, 0);
    ks.push(9, 0);
    ks.push(7, 0);

    cout << "Popped element from stack 2 is " << ks.pop(2) << endl;
    cout << "Popped element from stack 1 is " << ks.pop(1) << endl;
    cout << "Popped element from stack 0 is " << ks.pop(0) << endl;

    return 0;
}
```

10. STOCK SPAN PROBLEM

```
#include <bits/stdc++.h>
using namespace std;

void printSpan(int arr[],int n){

    stack<int> s ;
    s.push(0);
    cout<<1<<" ";
    for(int i=1;i<n;i++){
        while(s.empty()==false && arr[s.top()]<=arr[i]){
            s.pop();
        }
        int span=s.empty() ? i+1 : i-s.top();
        cout<<span<<" ";
        s.push(i);
    }
}

int main()
{
    int arr[]={18,12,13,14,11,16};
    int n=6;
    printSpan(arr,n);
    return 0;
}
```

11. PREVIOUS GREATER ELEMENT

```
#include <bits/stdc++.h>
using namespace std;

void printPrevGreater(int arr[],int n){
    stack<int>s;
    s.push(arr[0]);
```

```
    for(int i=0;i<n;i++){
        while(s.empty()==false && s.top()<=arr[i])
            s.pop();
        int pg=s.empty()?-1:s.top();
        cout<<pg<<" ";
        s.push(arr[i]);
    }
}

int main()
{
    int arr[]={20,30,10,5,15};
    int n=5;
    printPrevGreater(arr,n);
    return 0;
}
```

12. NEXT GREATER ELEMENT

```
#include <bits/stdc++.h>
using namespace std;

vector<int> nextGreater(int arr[],int n){
    vector<int> v;
    stack<int>s;
    s.push(arr[n-1]); v.push_back(-1);
    for(int i=n-2;i>=0;i--){
        while(s.empty()==false && s.top()<=arr[i])
            s.pop();
        int ng=s.empty()?-1:s.top();
        v.push_back(ng);
        s.push(arr[i]);
    }
    reverse(v.begin(),v.end());
```

```
    return v;
}

int main()
{
    int arr[]={5,15,10,8,6,12,9,18};
    int n=8;
    for(int x: nextGreater(arr,n)){
        cout<<x<< " ";
    }
    return 0;
}
```

13. LARGEST RECTANGULAR AREA IN HISTOGRAM

```
#include <bits/stdc++.h>
using namespace std;

int getMaxArea(int arr[],int n){
    int res=0;
    int ps[n],ns[n];

    stack <int> s;
    s.push(0);
    for(int i=0;i<n;i++){
        while(s.empty()==false && arr[s.top()]>=arr[i])
            s.pop();
        int pse=s.empty()?-1:s.top();
        ps[i]=pse;
        s.push(i);
    }

    while(s.empty()==false){
        s.pop();
```

```
    }

    s.push(n-1);
    for(int i=n-1;i>0;i--){
        while(s.empty()==false && arr[s.top()]>=arr[i])
            s.pop();
        int nse=s.empty()?n:s.top();
        ns[i]=nse;
        s.push(i);
    }

    for(int i=0;i<n;i++){
        int curr=arr[i];
        curr+=(i-ps[i]-1)*arr[i];
        curr+=(ns[i]-i-1)*arr[i];
        res=max(res,curr);
    }
    return res;

}

int main()
{
    int arr[]={6,2,5,4,1,5,6};
    int n=7;
    cout<<"Maximum Area: "<<getMaxArea(arr,n);
    return 0;
}
```

EFFICIENT CODE

```
#include <bits/stdc++.h>
using namespace std;

int getMaxArea(int arr[],int n){
```

```
    stack <int> s;
    int res=0;
    int tp;
    int curr;

    for(int i=0;i<n;i++){
        while(s.empty()==false && arr[s.top()]>=arr[i]){
            tp=s.top();s.pop();
            curr=arr[tp]* (s.empty() ? i : i - s.top() - 1);
            res=max(res,curr);
        }
        s.push(i);
    }
    while(s.empty()==false){
        tp=s.top();s.pop();
        curr=arr[tp]* (s.empty() ? n : n - s.top() - 1);
        res=max(res,curr);
    }

    return res;

}

int main()
{
    int arr[]={6,2,5,4,1,5,6};
    int n=7;
    cout<<"Maximum Area: "<<getMaxArea(arr,n);
    return 0;
}
```

14. LARGEST RECTANGLE WITH ALL 1s

```
#include <bits/stdc++.h>
using namespace std;

#define R 4
#define C 4

int largestHist(int arr[],int n)
{
    stack<int> result;
    int top_val;
    int max_area = 0;
    int area = 0;
    int i = 0;
    while (i < n) {
        if (result.empty() || arr[result.top()] <= arr[i])
            result.push(i++);

        else {
            top_val = arr[result.top()];
            result.pop();
            area = top_val * i;

            if (!result.empty())
                area = top_val * (i - result.top() - 1);
            max_area = max(area, max_area);
        }
    }
    while (!result.empty()) {
        top_val = arr[result.top()];
        result.pop();
        area = top_val * i;
        if (!result.empty())
```

```
            area = top_val * (i - result.top() - 1);

        max_area = max(area, max_area);
    }
    return max_area;
}

int maxRectangle(int mat[][C])
{
    int res = largestHist(mat[0],C);

    for (int i = 1; i < R; i++) {

        for (int j = 0; j < C; j++)
            if (mat[i][j])
                mat[i][j] += mat[i - 1][j];

        res = max(res, largestHist(mat[i],C));
    }
        return res;
}

int main()
{
    int mat[][C] = {
        { 0, 1, 1, 0 },
        { 1, 1, 1, 1 },
        { 1, 1, 1, 1 },
        { 1, 1, 0, 0 },
    };

    cout << "Area of maximum rectangle is " << maxRectangle(mat);

    return 0;
}
```

15. STACK WITH GETMIN() IN O(1)

```
#include <bits/stdc++.h>
using namespace std;

struct MyStack {

   stack<int> ms;
   stack<int> as;

void push(int x) {

     if(ms.empty() ) {
        ms.push(x);as.push(x);return;
     }

     ms.push(x);

     if(as.top()>=ms.top())
       as.push(x);
  }

void pop() {

   if(as.top()==ms.top())
      as.pop();
   ms.pop();

  }

int top() {
    return ms.top();
  }
```

```
int getMin() {
      return as.top();
   }
};

int main()
{
    MyStack s;
    s.push(4);
    s.push(5);
    s.push(8);
    s.push(1);
    s.pop();

    cout<<" Minimum Element from Stack: " <<s.getMin();

    return 0;
}
```

16. GETMIN() IN O(1) SPACE

```
#include <bits/stdc++.h>
using namespace std;

struct MyStack {

    stack<int> s;
    int min;

void push(int x) {

      if(s.empty() ) {
         min=x;
         s.push(x);
```

```
      }
      else if(x<=min){
          s.push(2*x-min);
          min=x;
      }else{
      s.push(x);
      }
    }

int pop() {

    int t=s.top();s.pop();
    if(t<=min){
        int res=min;
        min=2*min-t;
        return res;
    }else{
        return t;
    }
   }

int top() {
    int t=s.top();
    return ((t<=min)? min : t);
   }

int getMin() {
      return min;
   }
};

int main()
{
    MyStack s;
    s.push(4);
```

```
    s.push(5);
    s.push(8);
    s.push(1);
    s.pop();

    cout<<" Minimum Element from Stack: " <<s.getMin();

    return 0;
}
```

17. INFIX TO POSTFIX

18. EVALUATION OF POSTFIX

19. INFIX TO PREFIX

20. EVALUATION OF PREFIX