EASY :

1. IMPLEMENT 2 STACK IN AN ARRAY

```
//Function to push an integer into the stack1
vector<int>st;
void twoStacks :: push1(int x)
{
   this->arr[++this->top1] = x;
}

//Function to push an integer into the stack2.
void twoStacks ::push2(int x)
{
   this->arr[--this->top2] = x;
}

//Function to remove an element from top of the stack1.
int twoStacks ::pop1()
{
    if(this->top1>=0)
    return this->arr[this->top1--];
    return -1;

}

//Function to remove an element from top of the stack2.
int twoStacks :: pop2()
{
   if(this->top2<this->size)
   return this->arr[this->top2++];
   return -1;
}
```

2. PARENTHESIS CHECKER

```
class Solution
{
   public:
   bool ispar(string x)
   {
      stack<int>st;
      string s=x;
      for(int i=0;i<s.size();i++)
      {
         if(s[i]=='('||s[i]=='{'||s[i]=='[')
         st.push(s[i]);
         else if(st.size()==0)
         return false;
         else if(s[i]==')'&&st.top()=='(')
         st.pop();
         else if(s[i]=='}'&&st.top()=='{')
```

```
            st.pop();
            else if(s[i]==']'&&st.top()=='[')
            st.pop();

            else
            return false;

        }
        if(st.size()==0)
        return true;
        else
        return false;
    }

};
```

3. Length of Longest valid Substring

```
class Solution {
  public:
    int findMaxLen(string s) {
        // code here
        int ans=0;
        int n=s.length();
        int left=0;
        int right=0;
        for(int i=0;i<n;i++)
        {
            if(s[i]=='(')
            left++;
            else
            right++;
            if(left==right)
            ans=max(ans,2*right);
            else if(right>left)
            {
                left=0;
                right=0;
            }
        }

        left=0;
        right=0;

        for(int i=n-1;i>=0;i--)
        {
            if(s[i]=='(')
            left++;
            else
```

```
            right++;
            if(left==right)
            ans=max(ans,2*right);
            else if(left>right)
            {
               left=0;
               right=0;
            }
        }
        return ans;
    }
};
```

4. Implement Stack using queue

```
void QueueStack :: push(int x)
{
    q1.push(x);
}

int QueueStack :: pop()
{
        if(q1.empty())
        return -1;
        while(q1.size() != 1)
        {
            q2.push(q1.front());
            q1.pop();
        }

        int ans=q1.front();
        q1.pop();

        while(!q2.empty())
        {
            q1.push(q2.front());
            q2.pop();
        }
        return ans;

}
```

MEDIUM :

1. SPECIAL STACK

```
void push(stack<int>& s, int a)
{
        s.push(a);
}

bool isFull(stack<int>& s,int n)
{
        return s.size()==n;
}

bool isEmpty(stack<int>& s)
{
  return s.empty();
}

int pop(stack<int>& s)
{
  if(!s.empty())
        {
          int k=s.top();
          s.pop();
          return k;
        }
        return -1;
}

int getMin(stack<int>& s)
{
        int top=s.top();
        int mini=s.top();
        if(s.size()>1)
        {
          s.pop();
          mini=min(mini, getMin(s));
        }
        s.push(top);
        return mini;
}
```

2. Next greater number

```
class Solution
{
  public:
  vector<long long> nextLargerElement(vector<long long> arr, int n){
    stack<long long int> st;
    st.push(arr[n-1]);
    vector<long long int>v;
    v.push_back(-1);
    for(int i=n-2;i>=0;i--)
    {
      long long int tp=st.top();
```

```
            if(tp>arr[i])
            {
               v.push_back(tp);
               st.push(arr[i]);
            }
            else
            {
               while(!st.empty() && arr[i]>tp)
               {
                 st.pop();
                 if(st.empty()==false)
                 tp=st.top();
               }
               if(st.empty()==true)
               {
                  v.push_back(-1);
                  st.push(arr[i]);
               }
               else
               {
                   v.push_back(tp);
                   st.push(arr[i]);
               }
            }
         }
         reverse(v.begin(),v.end());
         return v;
     }
};
```

3. The celebrity problem

```
int celebrity(vector<vector<int> >& M, int n)
   {
      stack<int> st;

      for(int i=0;i<n;i++)
      st.push(i);

      while(st.size()>1)
      {
         int r=st.top();
         st.pop();
         int c=st.top();
         st.pop();

         if(M[r][c]==1)
         {
            if(M[c][r]==0)
            st.push(c);
```

```
        }
        else
        {
            if(M[c][r]==1)
            st.push(r);
        }
    }
    if(st.size()==1)
    {
      for(int i=0;i<n;i++)
      {
          if((M[i][st.top()]==0 || M[st.top()][i]==1) && st.top()!=i)
         {
             st.pop();
             return -1;
         }
      }
      return st.top();
     }
    return -1;
  }
```

4. Evaluation of postfix expression

```
class Solution
{
   public:
   int evaluatePostfix(string S)
   {
      string exp=S;
      stack<int>st;
      int a, b;
      for(int i=0; i<exp.length(); i++)
      {
            if(exp[i]>='0' && exp[i]<='9')
            st.push(exp[i]-'0');
            else
            {
               a = st.top();
               st.pop();
               b = st.top();
               st.pop();

               if(exp[i]=='+')
               st.push(b+a);
               else if(exp[i]=='-')
               st.push(b-a);
               else if(exp[i]=='*')
```

```
                st.push(b*a);
                else if(exp[i]=='/')
                st.push(b/a);
            }
        }
        return st.top();
    }
};
```

5. Sort a Stack using recursion

```
void SortedStack :: sort()
{
  stack<int>st;
  priority_queue<int>q;

  while(!s.empty())
  {
    q.push(s.top());
    s.pop();
  }
  while(!q.empty())
  {
    st.push(q.top());
    q.pop();
  }
  while(!st.empty())
  {

    s.push(st.top());
    st.pop();
  }
}
```

6. Merge overlapping Intervals

```
class Solution {
public:
    vector<vector<int>> merge(vector<vector<int>>& intervals)
    {
        int n=intervals.size();
        if(n<=1)
        return intervals;
        sort(intervals.begin(),intervals.end());

        vector<int>ans1=intervals[0];
        vector<vector<int>>ans;
```

```
        for(auto it:intervals)
        {

            if(ans1[1]>=it[0])
            ans1[1]=max(ans1[1],it[1]);
            else
            {
                ans.push_back(ans1);
                ans1=it;
            }
        }

        ans.push_back(ans1);

        return ans;

    }
};
```

7. Largest rectangular Area in Histogram

```
class Solution
{
    public:
    long long getMaxArea(long long arr[], int n)
    {
        // Your code here
       arr[n] = 0;
       stack<long long int> st;
       st.push(-1);
       long long int ans = 0;
       for(int i=0;i<=n;i++)
       {
          while(st.size()>1 && arr[st.top()]>=arr[i])
          {
             long long int ind = st.top();
             st.pop();
             ans = max(ans,arr[ind]*(i-st.top()-1));
          }
          st.push(i);
       }
       return ans;
    }
};
```