ARRAY AND VECTOR

GFG

Q1)taking input and output

Well there are several ways of taking inputs in a vector

Method1

```
#include <bits/stdc++.h>
using namespace std;
void printvec(vector<int>&v){
  for(int i=0;i<v.size();i++){
    cout<<v[i];
  }
  cout<<" " <<endl;

  for(auto it=v.begin();it!=v.end();it++){
    cout<<(*it);
  }
  cout<<" " <<endl;
  for(auto &value:v){
    cout<<value;
  }
}
int main() {
int a,b,c,d,m,n;
cin>>n;
vector<int>v;
int temp;
for(int i=0;i<n;i++){
  cin>>temp;
  v.push_back(temp);
}
printvec(v);
return 0;
}
```

Note the input can be taken in the array way cin>>a[i] too

Q2)taking a copy

```
#include <bits/stdc++.h>
using namespace std;
void printvec(vector<int>&v){
  for(int i=0;i<v.size();i++){
    cout<<v[i];
  }
}
int main() {
int a,b,c,d,m,n;
cin>>n;
vector<int>v;

int temp;
for(int i=0;i<n;i++){
  cin>>temp;
  v.push_back(temp);
}
vector<int>v2=v;
v2.push_back(12);
printvec(v2);
return 0;
}
```

Q3)function of a vector

So the vector has many inbuilt functions that can be defined as in the given example

```
#include <bits/stdc++.h>
using namespace std;
void min(vector<int>&v){
   int min=*min_element(v.begin(),v.end());
   cout<<min<<endl;
}
void max(vector<int>&v){
    int max=*max_element(v.begin(),v.end());
    cout<<max<<endl;
}
void sum(vector<int>&v){
      int sum=accumulate(v.begin(),v.end(),0);
      cout<<sum<<endl;
}
void count(vector<int>&v){
     int ct=count(v.begin(),v.end(),2);
     cout<<ct<<endl;
      int dt=count(v.begin()+1,v.end(),1);
     cout<<dt<<endl;//edit it you can  apply it to
any thing
}
void reverse(vector<int>&v){
       reverse(v.begin(),v.end());
     for(auto val:v){
        cout<<val;
  }
  cout<<endl;
}
void find(vector<int>&v){
  auto it=find(v.begin(),v.end(),2);
      if(it!=v.end()){
         cout<<"found";
      }
      else{
         cout<<"not found";
      }


     }
int main() {
int a,b,c,d,e,f,m,n;
cin>>n;
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
min(v);
max(v);
sum(v);
count(v);
reverse(v);
find(v);
return 0;
}
```

Note (refer)

https://www.geeksforgeeks.org/vector-in-cpp-stl/

Q4)defining algo in vectors

Q5)searching

```
#include <bits/stdc++.h>
using namespace std;
void search(vector<int>&v,int m){
  for(auto it=v.begin();it!=v.end();it++){
      if(*it==m){
         cout<<*it<<"found";
      }
  }
}
int main() {
int a,b,c,d,e,f,m,n;
cin>>m;
cin>>n;
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
search(v,m);
return 0;
}
```

Q6)insert

```
#include <bits/stdc++.h>
using namespace std;
void insert(vector<int>&v,int m,int p,int n){
   int pos=p-1;
   int start=n-1;
   v.resize(n+1);
   for(auto it=start;it>=pos;it--){
    v[it+1]=v[it];//shifting
   }
   v[pos]=m;
   for(auto it=v.begin();it<v.end();it++){
      cout<<*it;
   }
   cout<<endl;
   //another way of printing
   for(auto it=0;it<v.size();it++){
      cout<<v[it];//no pointers
   }
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>m;//number to be inserted
cin>>p;//position to be inserted
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
insert(v,m,p,n);
return 0;
}
```

Stl method

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

int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
v.insert(v.begin()+1,3);
for(auto &value:v){
   cout<<value;
}
return 0;

}
```

Q7)deletion

STL method: (https://www.geeksforgeeks.org/vector-erase-and-clear-in-cpp/)

There is basically two types of function existing in the stl function one is erase and the other is clear function

Ex-just v.clear() will erase it all

Erase function in the other hand can be used to specify which part to delete

Code

```
#include <bits/stdc++.h>

using namespace std;


int main() {

int a,b,c,d,e,f,m,n,p;

cin>>n;//number of elements

vector<int>v;

int temp;

for(int i=0;i<n;i++){

    cin>>temp;

    v.push_back(temp);

}

vector<int>::iterator it1,it2;

it1=v.begin();

it2=v.end();

it2--;//defining the range

it2--;

v.erase(it1,it2);

for(auto &value:v){

    cout<<value;

}



return 0;

}
```

Normal method:

```
#include <bits/stdc++.h>

using namespace std;


int sea(vector<int>&v,int m){

    int begin=0;

    vector<int>::iterator it;

    for(it=v.begin();it!=v.end();it++){

        if(*it==m){

            return begin;

            exit(0);

        }

        begin++;

    }

    return 0;

  }


    void del(vector<int>&v,int m,int n){

      auto it=find(v.begin(),v.end(),m);

       if(it!=v.end()){

         cout<<"element found to be deleted"<<endl;

       }

       else{

          cout<<"not found";

          exit(0);

       }

    int j=sea(v,m);

    for(auto it=j;it!=n-1;it++){
```

```
    v[it]=v[it+1];
  }
  v.resize(n-1);
  for(auto &value:v){
    cout<<value;
  }
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>m;//element to be deleted
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
  cin>>temp;
  v.push_back(temp);
}
del(v,m,n);

return 0;
}
```

Q8)largest element

STL:

We have already seen int *max_element work

FUNCTION:

```
#include <bits/stdc++.h>
using namespace std;
void largest(vector<int>&v,int n){
  int res=0;
  for(auto it=0;it<n;it++){
    if(v[it]>v[res]){
      res=it;
    }
  }
  cout<<"largest"<<v[res];
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
  cin>>temp;
  v.push_back(temp);
}
largest(v,n);

return 0;
}
```

Q9)second largest

```
#include <bits/stdc++.h>
using namespace std;
void seclarg(vector<int>&v,int n){
  int res=-1,largest=0;
  for(auto it=1;it<v.size();it++){
    if(v[it]>v[largest]){
      res=largest;
      largest=it;
    }else if(v[it]!=v[largest]){
      if(res==-1 || v[res]>v[it] ){
        res=it;
      }
    }
  }
```

```
    cout<<"largest element"<<v[largest]<<endl;
    cout<<"second largest"<<v[res];
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
seclarg(v,n);

return 0;
}
```

Q10)check whether sorted or not

```
#include <bits/stdc++.h>
using namespace std;
void sorted(vector<int>&v,int n){
    int res=0;
    for(auto it=1;it<v.size();it++){
        if(v[it]<v[res]){
            cout<<"not sorted"<<endl;
            exit(0);
        }
    }
    cout<<"sorted"<<endl;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
sorted(v,n);

return 0;
}
```

Q11)reverse an array

STL METHOD

We have already seen reverse(v.begin(),v.end()) working

NORMAL

```
#include <bits/stdc++.h>
using namespace std;
void reverse(vector<int>&v,int n){
    int high=n-1;
    int low=0;
    int temp;
    while(low<high){
        temp=v[low];
        v[low]=v[high];
        v[high]=temp;
        high--;
        low++;
    }
    for(auto it=0;it<n;it++){
        cout<<v[it];
    }
}
int main() {
```

```
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
reverse(v,n);


return 0;
}
```

Q12)bubble sort

```
#include <bits/stdc++.h>
using namespace std;
void bubble(vector<int>&v,int n){
   for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
          if(v[j]>v[j+1]){
             swap(v[j],v[j+1]);
          }
        }
     }
   for(auto &value:v){
      cout<<value;
   }
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
bubble(v,n);


return 0;
}
```

Q13)remove duplicates (https://www.geeksforgeeks.org/vector-in-cpp-stl/**)**

```
#include <bits/stdc++.h>
using namespace std;
void removedupli(vector<int>&v,int n){
 int count=0,res=1;
 sort(v.begin(),v.end());
 for(auto &value:v){
    cout<<value;
 }
 cout<<endl;
 for(auto it=1;it<n;it++){
    if(v[res-1]!=v[it]){
       v[res]=v[it];
       res++;
    }
 }
 for(auto it=0;it<res;it++){
   cout<<v[it];
}
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
removedupli(v,n);



return 0;
}
```

Q14)move all the zeros to the end

```
#include <bits/stdc++.h>
using namespace std;
void moveto(vector<int>&v,int n){
   int count=0;
   for(auto it=0;it<n;it++){
       if(v[it]!=0){
           swap(v[it],v[count]);
           count++;
       }
   }
   for(auto &value:v){
       cout<<value;
   }
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
moveto(v,n);



return 0;
}
```

Q15)left rotate by 1

```
#include <bits/stdc++.h>
using namespace std;
void moveto(vector<int>&v,int n){
  int temp=v[0];
  for(auto it=1;it<n;it++){
     v[it-1]=v[it];
  }
  v[n-1]=temp;
  for(auto &value:v){
      cout<<value;
  }

}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
moveto(v,n);



return 0;
}
```

Q16)left rotate by d

Naïve:

the naïve method will involve the calling one rotation d times through the for loop

pro

```
#include <bits/stdc++.h>

using namespace std;

void movetod(vector<int>&v,int n,int d){

   vector<int>v2(d);

  for(auto it=0;it<d;it++){

     v2[it]=v[it];

  }


  for(auto it=d;it<n;it++){

    v[it-d]=v[it];

  }

  for(auto it=0;it<d;it++){

     v[n-d+it]=v2[it];

  }


  for(auto &value:v){

     cout<<value;

  }


}

int main() {

int a,b,c,d,e,f,m,n,p;

cin>>d;//d positions moved

cin>>n;//number of elements

vector<int>v;

int temp;

for(int i=0;i<n;i++){

   cin>>temp;

   v.push_back(temp);

}

movetod(v,n,d);



return 0;

}
```

Q17)leader of a array

Leader of an array means nothing is greater than the element in the right of it

Naïve

```
#include <bits/stdc++.h>

using namespace std;

void leader(vector<int>&v,int n){

   for(auto it=0;it<n;it++){

       bool flag=false;

       for(int j=it+1;j<n;j++){

          if(v[it]<v[j]){

             flag= true;

             break;

          }

       }

       if(flag==false){//remember double=

          cout<<v[it];

       }

   }


}

int main() {

int a,b,c,d,e,f,m,n,p;

cin>>n;//number of elements

vector<int>v;

int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
leader(v,n);



return 0;
}
```

pro

```
#include <bits/stdc++.h>
using namespace std;
void leader(vector<int>&v,int n){
   int curr=n-1;
   cout<<v[curr];
   for(auto it=n-2;it>=0;it--){
      if(v[it]>v[curr]){
         cout<<v[it];
      }
   }

}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
leader(v,n);

return 0;
}
```

Q18)maximum diff(j>i)

Naïve

```
#include <bits/stdc++.h>
using namespace std;
void maxval(vector<int>&v,int n){
  int res=v[1]-v[0];
  for(auto it=0;it<n;it++){
     for(auto jt=1;jt<n;jt++){
        res=max(res,v[jt]-v[it]);
     }
  }
  cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
maxval(v,n);
return 0;
}
```

pro

```
#include <bits/stdc++.h>
using namespace std;
void maxval(vector<int>&v,int n){
  int mini=v[0];
```

```
 int res=v[1]-v[0];
 for(auto it=1;it<n;it++){
    res=max(res,v[it]-mini);
    mini=min(mini,v[it]);
 }
 cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
maxval(v,n);
return 0;
}
```

Q19)frequency of a array

```
#include <bits/stdc++.h>
using namespace std;
void freq(vector<int>&v,int n){
 sort(v.begin(),v.end());
   for(auto &value:v){
    cout<<value;
 }
 cout<<endl;
 int count=1;
 for(auto it=1;it<n;it++){
    if(v[it]==v[it-1]){
       count++;
    }else{
       cout<<count<<" "<<v[it-1];
       cout<<endl;
       count=1;

    }
  }//last elemet pending(guess why)
  cout<<v[n-1]<<count;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
freq(v,n);
return 0;
}
```

Q20)stock buy sell

This is popular interview problem one is the recursive way to do it and the other way is through loops

```
#include <bits/stdc++.h>
using namespace std;
void stockbuy(vector<int>&v,int n){
 int profit=0;
 for(auto it=1;it<n;it++){
    if(v[it]>v[it-1]){
       profit=profit+(v[it]-v[it-1]);
    }
 }
 cout<<profit;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
stockbuy(v,n);
return 0;
}
```

Q21)trapping rain water

this too is a popular interview problem

naïve

```
#include <bits/stdc++.h>
using namespace std;
void trapping(vector<int>&v,int n){
    int res=0;
    for(auto i=1;i<n-1;i++){
        int lmax=v[i];
        for(auto j=0;j<i;j++){
            lmax=max(lmax,v[j]);
        }
        int rmax=v[i];
        for(auto j=i+1;j<n;j++){
            rmax=max(rmax,v[j]);
        }
        res=res+min(lmax,rmax)-v[i];
    }
    cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
trapping(v,n);
return 0;
}
```

Pro

In this we pre compute the array and then solve

```
#include <bits/stdc++.h>
using namespace std;
void trapping(vector<int>&v,int n){
    int lmax[n],rmax[n],res;
    lmax[0]=v[0];
    for(int i=1;i<n;i++){
        lmax[i]=max(v[i],lmax[i-1]);
    }
    rmax[0]=v[n-1];
    for(int i=n-2;i>=0;i--){
        rmax[i]=max(v[i],rmax[i+1]);
    }
    for(int i=1;i<n-1;i++){
        res=res+(min(lmax[i],rmax[i])-v[i]);
    }
    cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
trapping(v,n);
return 0;
}
```

Q22)maximum consecutive 1 in binary array

So binary can be represented in either 0 and 1

```
#include <bits/stdc++.h>
using namespace std;
void maxcon(vector<int>&v,int n){
int count=0,res=0;
for(auto it=0;it<n;it++){
   if(v[it]==1){
      count=count+1;
      res=max(res,count);
   }else{
      count=0;
   }
}
cout<<res;

}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
maxcon(v,n);
return 0;
}
```

Q23)maximum sum sub array

Subarrays are basically contigious elements picked from the array like{1,2,3} are {1},{2},{3},{1,2},{2,3},{1,3}{1,2,3}

Naïve

```
#include <bits/stdc++.h>
using namespace std;
void maxsum(vector<int>&v,int n){
int res=v[0];
for(auto it=0;it<n;it++){
   int curr=0;
   for(int j=it;j<n;j++){
      curr=curr+v[it];
      res=max(res,curr);


   }
}
cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
maxsum(v,n);
return 0;
}
```

Pro

```
#include <bits/stdc++.h>
using namespace std;
void maxsum(vector<int>&v,int n){
    int res=v[0],maxend=v[0];
    for(auto it=1;it<n;it++){
        maxend=max(maxend+v[it],v[it]);
        res=max(res,maxend);
    }
    cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
maxsum(v,n);
return 0;
}
```

Q24)longest even odd sub array

The longest even off sub array that the odd and the even numbers are in continuous nature

Naïve

```
#include <bits/stdc++.h>
using namespace std;
void eveodd(vector<int>&v,int n){
    int res=1;
    for(int i=0;i<n;i++){
        int count=1;
        for(int j=i+1;j<n;j++){
            if((v[j]%2==0 && v[j-1]%2!=0) ||(v[j-
1]%2==0 && v[j]%2!=0)){
                count++;
                res=max(res,count);
            }else{
                count=1;
            }

        }
    }
    cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
eveodd(v,n);
return 0;
}
```

Pro(kadanes algo)

```
#include <bits/stdc++.h>
```

```
using namespace std;
void eveodd(vector<int>&v,int n){
   int res=1;
   int count=1;
   for(int j=1;j<n;j++){
       if((v[j]%2==0 && v[j-1]%2!=0) ||(v[j-1]%2==0
&& v[j]%2!=0)){
             count++;
             res=max(res,count);
       }else{
             count=1;
          }

       }
       cout<<res;
   }



int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
eveodd(v,n);
return 0;
}
```

Q25)maximum circular sub array sum

The difference between normal subarray and the circular subarray is like ex {10,5,-5} here the circular subarray is like {-5,10,5} where as the normal subarray is {10,5,-5} only

Naïve

```
#include <bits/stdc++.h>
using namespace std;
void maxcircle(vector<int>&v,int n){
  int res=v[0];
  for(auto it=0;it<n;it++){
     int curmax=v[it];
     int cursum=v[it];
     for(auto vt=1;vt<n;vt++){
         int idx=(it+vt)%n;
         cursum=cursum+v[idx];
         curmax=max(curmax,cursum);
    }
    res=max(res,curmax);
  }
  cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
maxcircle(v,n);
return 0;
}
```

Pro

to find the solution in o(n) we first find the sum of normal subarray (by kadanes algo) and then by normal circular sub array formula

```
#include <bits/stdc++.h>

using namespace std;

int normalsum(vector<int>&v,int n){

    int maxending=v[0],res=v[0];

    for(auto i=1;i<n;i++){

        maxending=max(maxending+v[i],v[i]);

        res=max(res,maxending);

    }

    return res;

}

void maxcircle(vector<int>&v,int n){

 int maxnormal=normalsum(v,n);

 int sum=0;

 if(maxnormal<0){

    cout<<maxnormal;

    exit(0);

 }

 for(auto i=0;i<n;i++){

    sum=sum+v[i];

    v[i]=-v[i];//inverting the elments

 }

 int maxcircle=sum+normalsum(v,n);

 int p=max(maxcircle,maxnormal);

 cout<<p;

}

int main() {

int a,b,c,d,e,f,m,n,p;

cin>>n;//number of elements

vector<int>v;

int temp;

for(int i=0;i<n;i++){

    cin>>temp;

    v.push_back(temp);

}

maxcircle(v,n);

return 0;

}
```

Q26)majority element

A element is said to be a majority element If it occurs more than n/2 times in a array

One method involves is through the 2 for loops and if they are equal we increase the count and store it ,it takes thus 0(n2) time but the method for O(n) is

```
 #include <bits/stdc++.h>

using namespace std;


void majelement(vector<int>&v,int n){

   int res=0,cur=1;

   for(auto i=1;i<n;i++){

      if(v[res]=v[i]){

         cur++;

      }else{

         cur--;

      }

      if(cur==0){

         cur=1;

         res=i;

      }

   }

   cur=0;

   for(int i=0;i<n;i++){

      if(v[res]==v[i]){//remember the ==

         cur++;

      }
```

```
  }
  if(cur>n/2){
     cout<<v[res]<<"majority element"<<endl;
  }else{
     cout<<"no majority found";
  }
}
int main() {
int a,b,c,d,e,f,m,n,p;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
majelement(v,n);
return 0;
}
```

Q27)apply sliding window technique to find the maximum sum of k consecutive elements

Normal method

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

void slidingwindow(vector<int>&v,int n,int k){
 int res=0;
 for(auto i=0;i<n-k+1;i++){
     int sum=0;
     for(auto j=i;j<k+i;j++){
        sum=sum+v[j];//you can take j=0 and allign
        //v[i+j]=0
     }
     res=max(res,sum);
 }
 cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p,k;
cin>>n;//number of elements
cin>>k;//number of elements for sum
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
slidingwindow(v,n,k);
return 0;
}
```

The sliding window technique

This technique is wildly popular and often asked in the technical interviews

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

void slidingwindow(vector<int>&v,int n,int k){
 int curr=0;
 for(auto i=0;i<k;i++){
     curr=curr+v[i];
  }//precomputing the kth elements
 //now we slide the elemnets in as per and find
 int res =curr;
 for(auto i=k;i<n;i++){
     curr=curr+v[i]-v[i-k];
     res=max(res,curr);
 }
 cout<<res;
}
int main() {
int a,b,c,d,e,f,m,n,p,k;
cin>>n;//number of elements
cin>>k;//number of elements for sum
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
slidingwindow(v,n,k);
return 0;
}
```

Q28)maximum consecutive flips

Maximum consecutive flips means lets take a binary array of {1,1,0,0,0,1} here either we can flip the 0 or the 1 along side each other consecutively like three zeros can be flipped together in one go so that's the easier option to flip rather than the two 1

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

void maxflips(vector<int>&v,int n){
  for(auto i=1;i<n;i++){
     if(v[i-1]!=v[i]){
        if(v[i]!=v[0]){
        cout<<"from"<<i;
     }else{
        cout<<i-1<<endl;
     }
     if(v[n-1]!=v[0]){
        cout<<n-1<<endl;
     }
    }
  }
}
int main() {
int a,b,c,d,e,f,m,n,p,k;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
   cin>>temp;
   v.push_back(temp);
}
maxflips(v,n);
return 0;
}
```

Q29)subarray with given sum

In the naïve approach similarly we can traverse through the whole array and see if the sum matches in the two for loops and if it matches we return the value

Pro

In this method we use the sliding window technique

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

void givensum(vector<int>&v,int n,int sum){
   int curr=0,start=0;
 for(int i=0;i<n;i++){
    curr=curr+v[i];
    while(sum<curr){
        curr=curr-v[start];
        start++;
    }
    if(curr==sum){
        cout<<sum<<" "<<"thus found";
        exit(0);
    }
 }
 cout<<"not found";
}
int main() {
int a,b,c,d,e,f,m,n,p,k,sum;
cin>>n;//number of elements
cin>>sum;//sum of the element you want
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
givensum(v,n,sum);
return 0;
}
```

Q30) prefix sum

This is a popular interview problem often asked in interviews and a common question in the competitive programming

pro

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


void prefixsum(vector<int>&v,int n,int a,int b){
  for(auto i=1;i<n;i++){
      v[i]=v[i]+v[i-1];
  } //preprocessing the thing
  for(auto it=v.begin();it!=v.end();it++){
      cout<<*it;
  }
  cout<<endl;


  if(b==0){
      cout<<v[a];
  }else{
      cout<<v[a]-v[b-1];
  }


}
int main() {
int a,b,c,d,e,f,m,n,p,k,sum;
cin>>n;//number of elements
cin>>a>>b;//range of sum
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
prefixsum(v,n,a,b);
return 0;
}
```

Q31)Equilibrium point

A point is said to be a equilibrium point if the sum and before and after is same

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


void equili(vector<int>&v,int n){
   int res=0;
   for(auto i=0;i<n;i++){
       res=res+v[i];
   }
  int sum=0;
  for(auto i=0;i<n;i++){
     res=res-v[i];
      if(sum==res){
          cout<<"sum found"<<sum;
          exit(0);
      }
       sum=sum+v[i];
  }

}
int main() {
int a,b,c,d,e,f,m,n,p,k,sum;
cin>>n;//number of elements
vector<int>v;
int temp;
for(int i=0;i<n;i++){
    cin>>temp;
    v.push_back(temp);
}
equili(v,n);
return 0;
}
```