STRINGS

1)basics c++ strings in usage and function

```
#include <bits/stdc++.h>
using namespace std;
int main() {
string s="helloworld";
cout<<s.length()<<endl;
s=s+" hellu";
cout<<s<<endl;
cout<<s.find("hel");
}
```

2)usage of the find function

Note:- the previous time what we saw is that the find function returns only the index now the real code

```
#include <bits/stdc++.h>
using namespace std;
int main() {
string input="hello world";
string str;
getline(cin,str);
int index=input.find(str);
if(index==-1){
   cout<<"string not found"<<endl;
}
if(index!=-1){
   cout<<"first occ"<<index<<endl;
}
index=input.find(str,index+1);
if(index!=-1){
   cout<<"secomd  occ"<<index<<endl;
}
}
```

Q3)using vectors take the input of the string

```
#include <bits/stdc++.h>
using namespace std;
int main() {
 int n;
 cin>>n;
 vector<string>arr;
 string temp;
for (int i=0;i<n;i++){
    cin>>temp;
    arr.push_back(temp);
 }
 for(string x:arr){
    cout<<x<<endl;
 }
}
```

Q4)lets code now about the palindrome

Note:- the word is said to be a palindrome if its reversed function is also the same like abcda

First lets understand how Boolean function works.when we pass a function through Boolean it either returns true (as1) or false (as 0)

Ex→

```
#include <bits/stdc++.h>
using namespace std;
 bool isSingleDigit (int x) {
  return (x >= 0 && x < 10);
 }
  int main () {
   cout << isSingleDigit (2) << endl;
   bool bigFlag = !isSingleDigit (17);
   if(bigFlag==1){
    cout<<"hello";
  }
  }
```

Naïve method:-

```
#include <bits/stdc++.h>
using namespace std;
bool pal(string &str){
   string rev=str;
   reverse(rev.begin(),rev.end());
   return(rev==str);
}
int main(){
 string str;
 cin>>str;
 bool palindrome=pal(str);
  if(palindrome==1){
    cout<<"yes its a palindrome"<<endl;
   }else{
      cout<<"no"<<endl;
 }
}
```

Vector way→

```
#include <bits/stdc++.h>
using namespace std;
bool pal(vector<string>&str){
   vector<string>rev=str;
   reverse(rev.begin(),rev.end());
   return(rev==str);
}
int main(){
 int n;
 cin>>n;
 vector<string>str;
 string temp;
 for(int i=0;i<n;i++){
   cin>>temp;
   str.push_back(temp);
 }
  bool palindrome=pal(str);
  if(palindrome==1){
     cout<<"yes its a palindrome"<<endl;
  }else{
     cout<<"no"<<endl;
  }
 }
```

Pro method:-

```
#include <bits/stdc++.h>
using namespace std;
bool pal(string &str){
   int begin=0;
   int end=str.length()-1;
   while(begin<end){
      if(str[begin]!=str[end]){
         return false;
      }
      begin++;
      end--;
   }
   return true;
 }
int main(){
  string str;
 cin>>str;
   bool palindrome=pal(str);
   if(palindrome==1){
      cout<<"yes its a palindrome"<<endl;
   }else{
```

```
    cout<<"no"<<endl;
  }
  }
```

Q6)check whether string is a subsequence or not

Here subsequence means that if abc is there in the string its subsequence is in the line of characters cab is not but ab ac is .

Code

```
#include <bits/stdc++.h>
using namespace std;
bool sub(string &s1,string &s2){
   int m=s1.length();
   int n=s2.length();
   int j=0;
   for(int i=0;i<m && j<n;i++){
      if(s1[i]==s2[j]){
         j++;
      }
   }
   return(j==n);
}
int main(){
  string str1,str2;
  cin>>str1>>str2;
  bool flag=sub(str1,str2);
  if(flag==1){
     cout<<"yes";
  }else{
     cout<<"no";
  }
 }
```

Q7)check for the anagram

So basically anagram means that all the characters must be present in first array must be present in the second array.like listen and silent.

Naïve

```
#include <bits/stdc++.h>
using namespace std;
bool ana(string &s1,string &s2){
   int m=s1.length();
   int n=s2.length();
   if(m!=n){
      return false;
   }
   sort(s1.begin(),s1.end());
   sort(s2.begin(),s2.end());
   return(s1==s2);
}
int main(){
  string str1,str2;
  cin>>str1>>str2;
  bool flag=ana(str1,str2);
  if(flag==1){
     cout<<"yes";
  }else{
     cout<<"no";
  }
 }
```

Pro

```
#include <bits/stdc++.h>
using namespace std;
const int hellu=256;
bool ana(string &s1,string &s2){
   int m=s1.length();
   int n=s2.length();
```

```
  if(m!=n){
     return false;
  }
  int count[hellu]={0};
  for(int i=0;i<m;i++){
     count[s1[i]]++;
     count[s2[i]]--;
  }
  for(int i=0;i<hellu;i++){
     if(count[i]!=0){
        return false;
     }
  }
  return true;
}
int main(){
  string str1,str2;
  cin>>str1>>str2;
  bool flag=ana(str1,str2);
  if(flag==1){
     cout<<"yes";
  }else{
     cout<<"no";
  }
 }
```

Q8)print the frequencies of character(in sorted order)of ex→lower case alphabets

Naïve

```
#include <bits/stdc++.h>
using namespace std;
const int hellu=256;
void ana(string &s1){
   int count[hellu]={0};
   int j=s1.length();
   for(int i=0;i<j;i++){
      count[s1[i]]++;
   }
   for(int i=0;i<256;i++){
      if(count[i]>0)
      cout<<char(i)<<" "<<count[i]<<endl;
   }
}
int main(){
  string str1;
  cin>>str1;
  ana(str1);

 }
```

Pro

```
#include<bits/stdc++.h>
using namespace std;
const int hellu=256;
  void ana(string &s1,string &s2){
     int m=s1.length();
     int n=s2.length();
     if(m!=n){
        cout<<"not a anagram"<<endl;
        exit(0);
     }
     int arr[hellu]={0};
     for(int i=0;i<m;i++){
        arr[s1[i]]++;
        arr[s2[i]]--;
     }

     for(int i=0;i<hellu;i++){
```

```
        if(arr[i]>0){
            cout<<"not a anagram"<<endl;
            exit(0);
        }
      }
      cout<<"is a anagram"<<endl;
  }
int main(){
   string str1,str2;
   cin>>str1>>str2;
   ana(str1,str2);
}
```

Q9)leftmost repeating element

Naïve

```
#include<bits/stdc++.h>
using namespace std;
void leftmost(string &s1){
   for(int i=0;i<s1.length();i++){
      for(int j=i+1;j<s1.length();j++){
         if(s1[i]==s1[j]){
            cout<<s1[i]<<endl;
            exit(0);
         }
      }
   }
}
int main(){
   string str;
   cin>>str;
   leftmost(str);
}
```

pro

```
#include<bits/stdc++.h>
using namespace std;
const int hellu=256;
  void left(string &s1){
     int m=s1.length();
     int arr[hellu]={0};
     for(int i=0;i<m;i++){
        arr[s1[i]]++;
     }
     for(int i=0;i<hellu;i++){
        if(arr[i]>1){
          cout<<arr[i]<<char(i)<<endl;
          exit(0);
        }
     }
cout<<"no repeating elemeyts found"<<endl;
  }
int main(){
   string str1;
   cin>>str1;
   left(str1);
}
```

more pro

```
#include<bits/stdc++.h>
using namespace std;
const int hellu=256;
  void left(string &s1){
     int m=s1.length();
     int arr[hellu];
     int res=INT_MAX;
     fill(arr,arr+hellu,-1);
     for(int i=0;i<m;i++){
        int fi=arr[s1[i]];
        if(fi==-1){
```

```
            arr[s1[i]]=i;
        }else{
            res=min(res,fi);
            cout<<res<<s1[res];//note the character
array
            exit(0);
        }
    }
  }
int main(){
   string str1;
   cin>>str1;
   left(str1);
}
```

Q10)leftmost non repeating element

Naïve

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



int nonRep(string &str)
{
   for(int i=0;i<str.length();i++){
      bool flag=false;
      for(int j=0;j<str.length();j++){
         if(i!=j&&str[i]==str[j]){
            flag=true;
            break;
         }
      }
      if(flag==false)return i;
   }
   return -1;
}

int main()
{
   string str = "geeksforgeeks";
   cout<<"Index of leftmost non-repeating
element:"<<endl;
   cout<<nonRep(str)<<endl;


   return 0;
}
```

Pro

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

const int CHAR=256;
int nonRep(string &str)
{
   int fI[CHAR];
   fill(fI,fI+CHAR,-1);

   for(int i=0;i<str.length();i++){
      if(fI[str[i]]==-1)
      fI[str[i]]=i;
      else
      fI[str[i]]=-2;
   }
   int res=INT_MAX;
   for(int i=0;i<CHAR;i++){
      if(fI[i]>=0)res=min(res,fI[i]);
   }
   return (res==INT_MAX)?-1:res;
```

```
}


int main()
{
   string str = "geeksforgeeks";
   cout<<"Index of leftmost non-repeating element:"<<endl;
   cout<<nonRep(str)<<endl;


   return 0;
}
```

Q11)reverse the word in the string

Naïve

1)Create a stack

2)push words one by one

3)pop words and append

Pro

```
#include<bits/stdc++.h>
using namespace std;
void reverse(char str[],int low, int high){
   while(low<=high){
      swap(str[low],str[high]);
      low++;
      high--;
   }
}
void reverseWords(char str[],int n){
   int start=0;
   for(int end=0;end<n;end++){
      if(str[end]==' '){
         reverse(str,start,end-1);
         start=end+1;
      }
   }
   reverse(str,start,n-1);
   reverse(str,0,n-1);
}
int main(){
   string str1;
   getline(cin,str1);
   rev(str1);


}
```

Q12)naïve pattern searching

Naïve

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

void patSearchinng(string &txt,string &pat){
   int m=pat.length();
   int n=txt.length();
   for(int i=0;i<=(n-m);i++){
      int j;
      for(j=0;j<m;j++)
      if(pat[j]!=txt[i+j])
      break;

      if(j==m)
      cout<<i<<" ";
   }
}

int main()
{
   string txt = "ABCABCD";string pat="ABCD";
```

```
   cout<<"All index numbers where pattern
found:"<<" ";
   patSearchinng(txt,pat);

   return 0;
}
```

pro

Q13)check if strings have rotations

To check whether the string has rotations or not

Pro

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

void rotations(string &s1,string &s2){
   int m=s1.length();
   int n=s2.length();
   if(m!=n){
      cout<<"not a rotation"<<endl;
      exit(0);
   }
   string s3=s1+s2;
   int s5=s3.find(s2);
   if(s5==-1){
      cout<<"not a rotation"<<endl;
   }
   if(s5!=-1){
      cout<<"string is a rotation"<<endl;
   }

}
int main(){
   string str1,str2;
   cin>>str1>>str2;
   rotations(str1,str2);

}
```

Q14)anagram search

pro

Note:-try with spaces

```
#include<bits/stdc++.h>
using namespace std;
const int hima=256;
bool anag(string &s1,string &s2,int i){
   int arr[hima]={0};
   int y=s2.length();
   for(int j=0;j<y;j++){
      arr[s1[j]]++;
      arr[s1[i+j]]--;
   }
   for(int j=0;j<hima;j++){
      if(arr[j]!=0){
         return false;
      }
   }
   return true;
}

void anag(string &s1,string &s2){
   int m=s1.length();
   int n=s2.length();
   int x=m-n;
   for(int i=0;i<x;i++){
      if(anag(s1,s2,i)){
       cout<<"yes its in the anagram in the
search"<<endl;
       exit(0);
```

```
    }
  }
 cout<<"its not "<<endl;
  }
int main(){
  string str1,str2;
  cin>>str1>>str2;
  anag(str1,str2);

}
```

Q15)lexicographic rank

Lexicographic rank means string order with leftmost having the most value ex-bac string abc rank 1 least and cab most at 7

Pro

```
#include<bits/stdc++.h>
using namespace std;
const int hima=256;
int fact(int n){
  int hell=1;//note if 0 is the case add another if statement
  for(int i=1;i<=n;i++){
    hell=hell*i;
  }
  return hell;
}
void lex(string &s1){
  int res=1;
  int n=s1.length();
  int mul=fact(n);
  int count[hima]={0};
  for(int i=0;i<n;i++){
    count[s1[i]]++;
  }
  for(int i=1;i<hima;i++){
    count[i]=count[i]+count[i-1];
  }
  for(int i=0;i<n-1;i++){
    mul=mul/(n-i);
    res=res+count[s1[i]-1]*mul;

  for(int j=s1[i];j<hima;j++){
    count[j]--;
  }

 }
 cout<<res<<endl;
}
int main(){
  string str1;
  cin>>str1;
  lex(str1);

}
```

Q16)longest substring with distinct characters

Naïve

pro