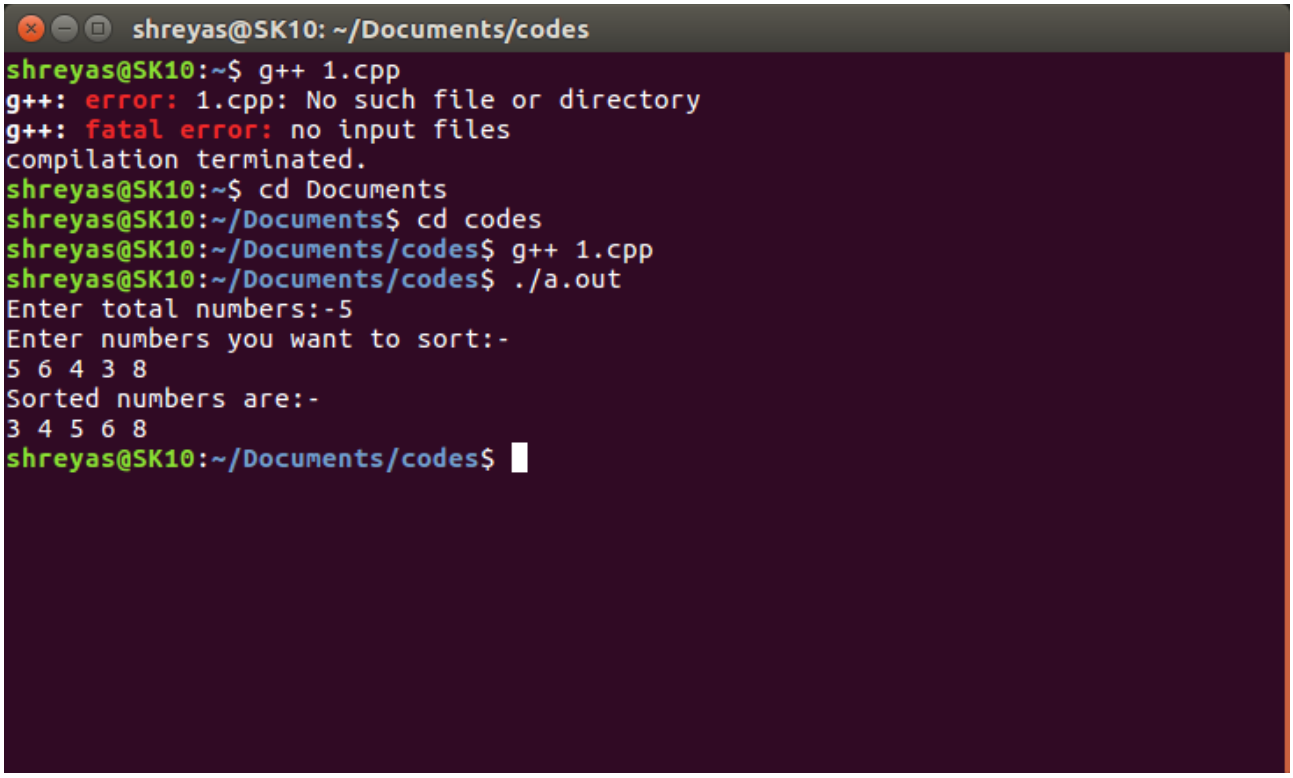
1.QUICKSORT

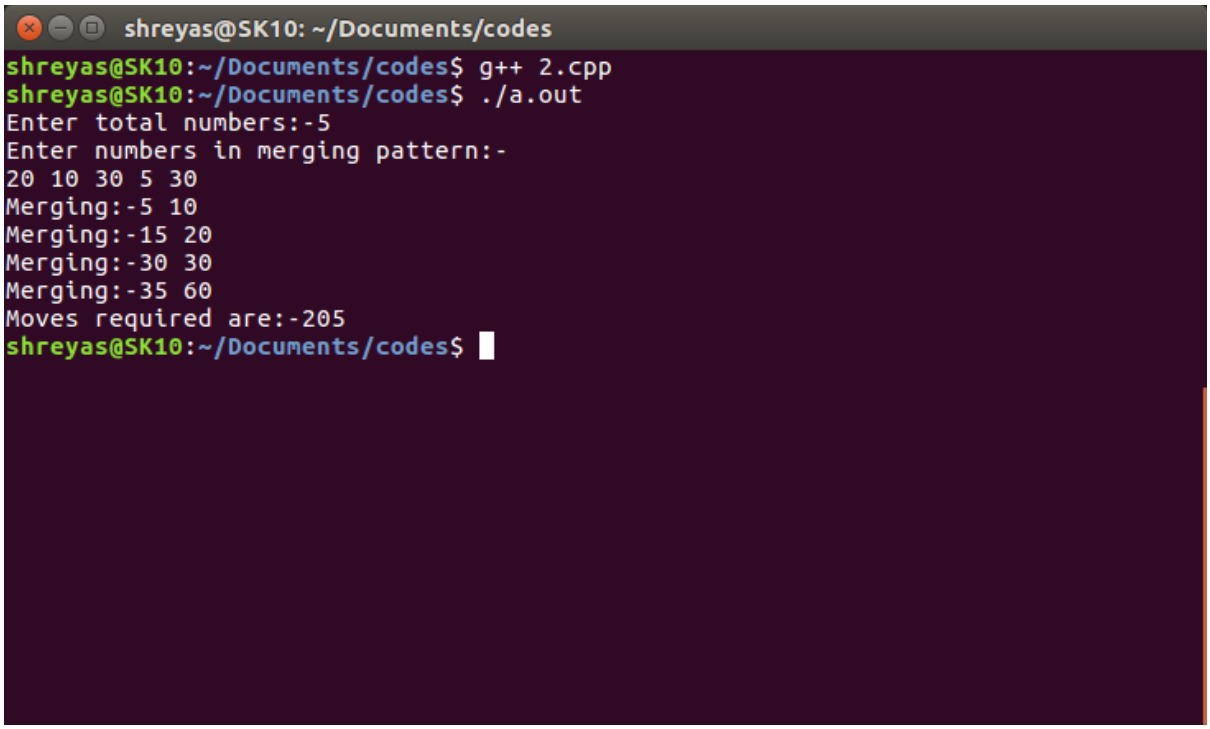
```
//Sorting
//QuickSort

#include<bits/stdc++.h>
using namespace std;
int part(int a[],int p,int r)
{
     int i=p-1,x=a[r];
     for(int j=p;j<r;j++)
     {
          if(a[j]<=x)
          {
               i++;
               int temp=a[i];
               a[i]=a[j];
               a[j]=temp;
          }
     }
     int temp=a[i+1];
     a[i+1]=a[r];
     a[r]=temp;
     return (i+1);
}

void q_sort(int a[],int p,int r)
{
     if(p<r)
     {
          int q=part(a,p,r);
          q_sort(a,p,q-1);
          q_sort(a,q,r);
     }
}

int main()
{
     int n;
     cout<<"Enter total numbers:-";
     cin>>n;
     int a[n];
     cout<<"Enter numbers you want to sort:-\n";
     for(int i=0;i<n;i++)
     {
          cin>>a[i];
     }
     q_sort(a,0,n-1);
     cout<<"Sorted numbers are:-\n";
     for(int i=0;i<n;i++)
     {
          cout<<a[i]<<" ";
     }
```

```
	cout<<endl;
	return 0;
}
```



2.OPTIMUM MERGE PATTERN

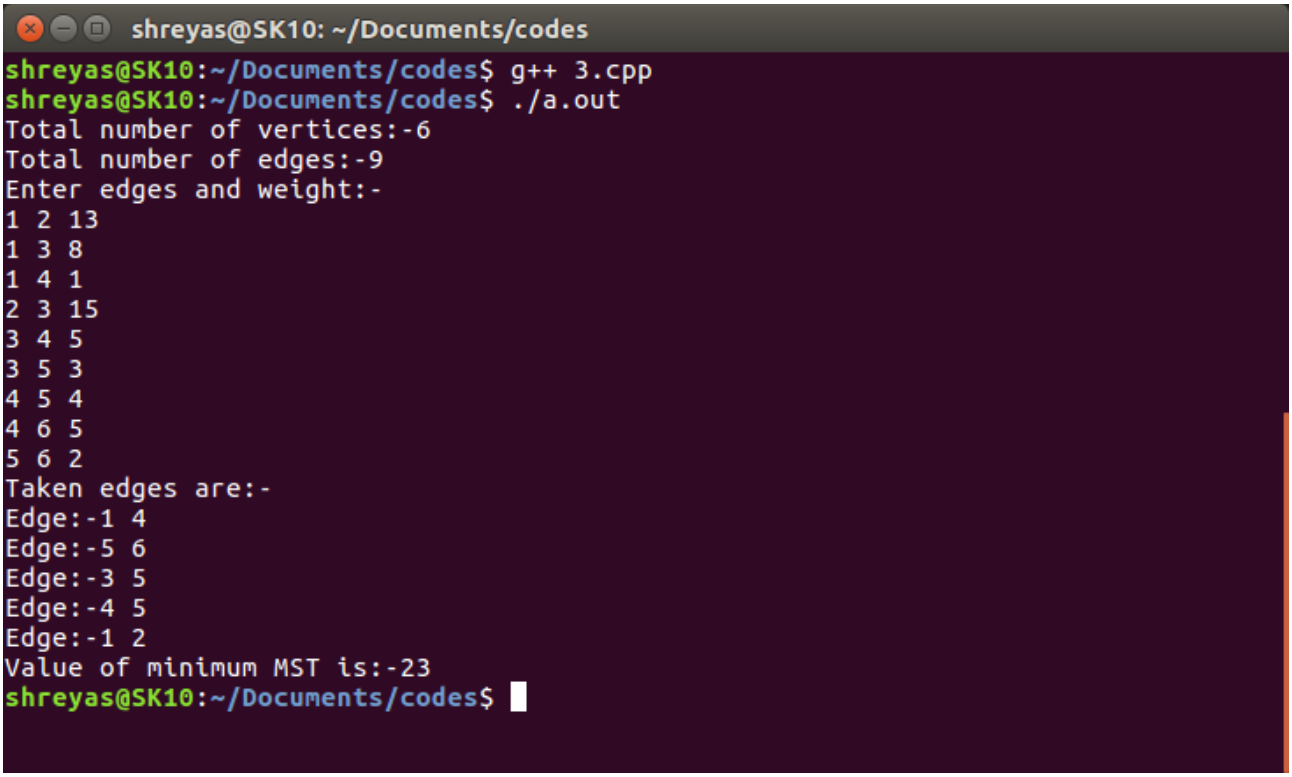
```
/Greedy Algorithm
//Optimum Merge Pattern


#include<bits/stdc++.h>
using namespace std;
int main()
{
     int n;
     cout<<"Enter total numbers:-";
     cin>>n;
     priority_queue <int, vector<int>, greater<int> > p_queue;
     cout<<"Enter numbers in merging pattern:-\n";
     for(int i=0;i<n;i++)
     {
          int temp;
          cin>>temp;
          p_queue.push(temp);
     }
     int sum=0;
     for(int i=0;i<n-1;i++)
     {
          int a=p_queue.top();
          p_queue.pop();
          int b=p_queue.top();
          p_queue.pop();
          sum+=(a+b);
          cout<<"Merging:-"<<a<<" "<<b<<endl;
          p_queue.push(a+b);
     }
     cout<<"Moves required are:-"<<sum;
     cout<<endl;
     return 0;
}
```


shreyas@SK10: ~/Documents/codes
shreyas@SK10:~/Documents/codes$ g++ 2.cpp
shreyas@SK10:~/Documents/codes$ ./a.out
Enter total numbers:-5
Enter numbers in merging pattern:-
20 10 30 5 30
Merging:-5 10
Merging:-15 20
Merging:-30 30
Merging:-35 60
Moves required are:-205
shreyas@SK10:~/Documents/codes$

3.KRUSKAL'S ALGORITHM

```
//Greedy Algorithm
//Kruskal's Algorithm

#include<bits/stdc++.h>
using namespace std;
int find(int parent[],int i)
{
     if(parent[i]==-1)
          return i;
     return find(parent,parent[i]);
}
void uni(int parent[],int i,int j)
{
     int x=find(parent,i);
     int y=find(parent,j);
     parent[x]=y;
}
int main()
{
     int n,m;
     cout<<"Total number of vertices:-";
    cin>>n;
    cout<<"Total number of edges:-";
    cin>>m;
    int parent[n+1];
     for(int i=1;i<=n;i++)
     {
          parent[i]=-1;
     }
    vector<pair<int,pair<int,int> > >weight;
    cout<<"Enter edges and weight:-\n";
    for(int i=0;i<m;i++)
    {
     int x,y,w;
     cin>>x>>y>>w;
     weight.push_back(make_pair(w,make_pair(x,y)));
    }
    sort(weight.begin(),weight.end());
    int sum=0;
    cout<<"Taken edges are:-\n";
     for(int i=0;i<m;i++)
     {
          int x=weight[i].second.first;
          int y=weight[i].second.second;
          int m=find(parent,x);
          int n=find(parent,y);
          if(m!=n)
          {
               uni(parent,m,n);
               sum+=(weight[i].first);
               cout<<"Edge:-"<<x<<" "<<y<<endl;
```

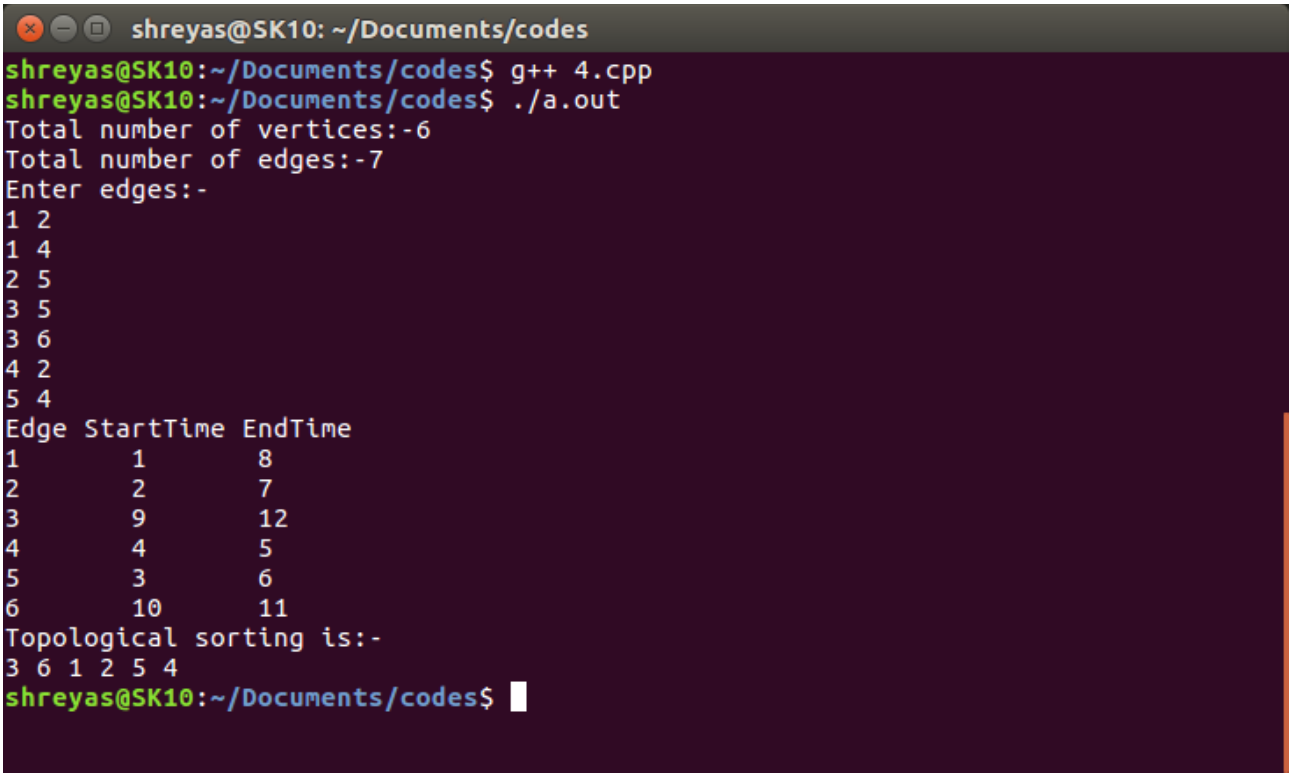
```
            }
        }
        cout<<"Value of minimum MST is:-"<<sum<<endl;
        return 0;
}
```



4.TOPOLOGICAL SORT

```
//Topological Sort using DFS

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

vector<int>a[100];
bool check[100];
int times=0;
int start[100],end[100];

void depth_f_search(int t)
{
     //cout<<t<<" ";
    check[t]=1;
    times++;
    start[t]=times;
    for(int i=0;i<a[t].size();i++)
    {
          if(check[a[t][i]]==0)
          depth_f_search(a[t][i]);
    }
    times++;
    end[t]=times;
}

int main()
{
     int n,m;
     cout<<"Total number of vertices:-";
    cin>>n;
    cout<<"Total number of edges:-";
    cin>>m;
    cout<<"Enter edges:-\n";
    for(int i=0;i<m;i++)
    {
     int x,y;
          cin>>x>>y;
          a[x].push_back(y);
    }
    for(int i=0;i<100;i++)
          check[i]=0;

    for(int i=1;i<=n;i++)
     {
          if(check[i]==0)
          {
             depth_f_search(i);
          }
    }
    cout<<"Edge StartTime EndTime\n";
```

```
    vector<pair<int,int> >v;
    for(int i=1;i<=n;i++)
    {
     v.push_back(make_pair(end[i],i));
     cout<<i<<"\t"<<start[i]<<"\t"<<end[i]<<endl;
    }
    sort(v.begin(),v.end());
    cout<<"Topological sorting is:-\n";
    for(int i=n-1;i>=0;i--)
    {
     cout<<v[i].second<<" ";
    }
    cout<<endl;
    return 0;
}
```



5.LONGEST COMMON SUBSEQUENCE

```
//Dynamic Programming
//Longest Common Subsequence

#include<bits/stdc++.h>
using namespace std;
char flag[100][100];
void print_sub(char a[100],int i,int j)
{
     if(i==0 || j==0)
          return;
     if(flag[i][j]=='q')
     {
          print_sub(a,i-1,j-1);
          cout<<a[i-1];
     }
     else if(flag[i][j]=='w')
          print_sub(a,i-1,j);
     else if(flag[i][j]=='a')
          print_sub(a,i,j-1);
}
int main()
{
     int n,m;
     char a[100],b[100];
     cout<<"Enter string1:-\n";
     cin>>a;
     cout<<"Enter string2:-\n";
     cin>>b;
     n=strlen(a);
     m=strlen(b);
     int check[n+1][m+1];

     for(int i=0;i<n+1;i++)
     {
          check[i][0]=0;
          flag[i][0]='0';
     }
     for(int i=1;i<m+1;i++)
     {
          check[0][i]=0;
          flag[0][i]='0';
     }
     for(int i=1;i<n+1;i++)
     {
          for(int j=1;j<m+1;j++)
          {
               if(a[i-1]==b[j-1])
               {
                    check[i][j]=check[i-1][j-1]+1;
                    flag[i][j]='q';
               }
```

```
                else
                {
                     check[i][j]=max(check[i-1][j],check[i][j-1]);
                     if(check[i-1][j]>=check[i][j-1])
                     {
                          flag[i][j]='w';
                     }
                     else
                     {
                          flag[i][j]='a';
                     }
                }
           }
     }
     cout<<"Maximum length of common subsequence is:-"<<check[n]
[m]<<endl;
     cout<<"Common subsequence is:-\n";
     print_sub(a,n,m);
     cout<<endl;
     return 0;
}
```