LAB Assignment - 4	
NAME: **Aashirwad Kumar**	Roll NUMBER: **IMH/10004/18**

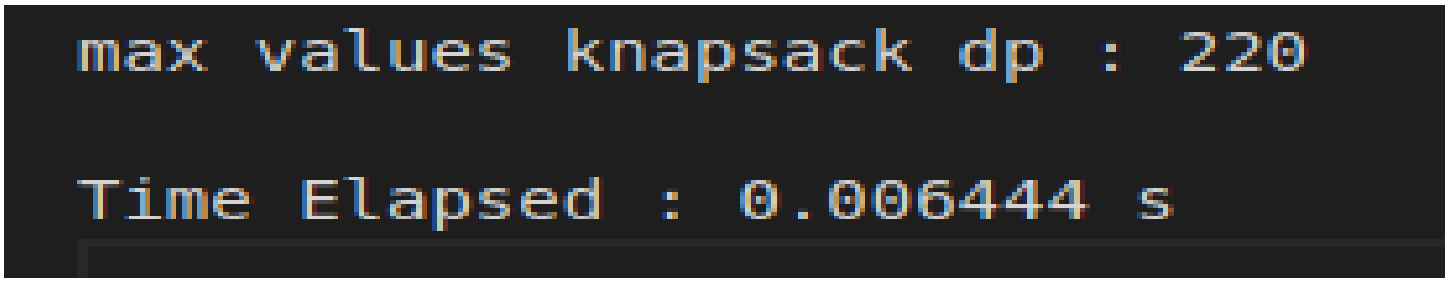
Question – 1 – a)	Implement the 0/1 Knapsack problem using Dynamic Programming method	

CODE

```
#include <bits/stdc++.h>
using namespace std;
#define fast ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
typedef long long ll;typedef long double ld;typedef pair<int,int> pii;
#define F first
#define S second
#define PB push_back
#define MP make_pair
const ll mod = 1e9+7, N = 2e6+7, M = 2e6+7, INF = INT_MAX/10;
ll powe(ll x, ll y){ x = x%mod, y=y%(mod-1);ll ans = 1;while(y>0){if (y&1)
{ans = (1ll * x * ans)%mod;}y>>=1;x = (1ll * x * x)%mod;}return ans;}

int findMax(vector<int>&items , vector<int>&weights , int capacity )
{
   int n=items.size();
   vector<vector<int>>dp(n+1,vector<int>(capacity+1,0));
   for(int i=1 ; i<=n ; i++)
   {
      int ite = items[i-1];
      int wt = weights[i-1];
      for(int j=1 ; j<=capacity ; j++)
      {
         if(wt<=j)
         {
            dp[i][j] = max(dp[i-1][j-wt]+ite , dp[i-1][j]);
         }
         else
         {
            dp[i][j] = dp[i-1][j];
         }
      }
   }
   return dp[n][capacity];
}
```

```
int main()
{
   vector<int> items{60,100,120};
   vector<int> weights{10,20,30};
   int capacity=50;
   int maxValues =findMax(items , weights , capacity);
   cout<<"max values knapsack dp : " <<maxValues;
   cout<<"\n";


   #ifndef ONLINE_JUDGE
      cout<<"\nTime Elapsed : " << 1.0*clock() / CLOCKS_PER_SEC << " s\
n";
   #endif

   return 0;
}
```

OUTPUT



Question -1 – b)	Implement the 0/1 Knapsack problem using Greedy method.	

CODE

```
#include <bits/stdc++.h>
using namespace std;
#define fast ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
typedef long long ll;typedef long double ld;typedef pair<int,int> pii;
#define F first
#define S second
#define PB push_back
#define MP make_pair
const ll mod = 1e9+7, N = 2e6+7, M = 2e6+7, INF = INT_MAX/10;
ll powe(ll x, ll y){ x = x%mod, y=y%(mod-1);ll ans = 1;while(y>0){if (y&1)
{ans = (1ll * x * ans)%mod;}y>>=1;x = (1ll * x * x)%mod;}return ans;}



void solve_prg(){

    int n;
    cin>>n;

    int x,y;
    vector<pair<int,int>> v;
    for(int i=0;i<n;i++){
        cin>>x>>y;
        v.push_back({x,y});
    }
    sort(v.begin(),v.end());

    int weight;
    cin>>weight;
    vector <pair<int,int>> bag;
    int wt_bag=weight;
    int val_bag=0;
    for(int i=0;i<n;i++){
        if(v[i].second<=wt_bag){
            bag.push_back(v[i]);
            wt_bag-=v[i].second;
            val_bag+=v[i].first;
        }
        else{
```

```
            for(int j=0;j<bag.size();j++){
                if(wt_bag+bag[j].second>=v[i].second && bag[j].first<v[i].first){
                    wt_bag+=bag[j].second;
                    wt_bag-=v[i].second;
                    val_bag-=bag[j].first;
                    val_bag+=v[i].first;
                    bag[j].second=v[i].second;
                    bag[j].first=v[i].first;
                    break;
                }
            }
        }
    }
    cout<<"Problem Optimum value "<<val_bag<<endl;

}

int main() {


   solve_prg();

   #ifndef ONLINE_JUDGE
       cout<<"\nTime Elapsed : " << 1.0*clock() / CLOCKS_PER_SEC << " s\
n";
    #endif
    return 0;
}
```

OUTPUT

input.txt

Current > input.txt

```
3
60 10
100 20
120 30
50
```

output.txt

Current > output.txt

```
Problem Optimum value 220

Time Elapsed : 0.006863 s

```

Question - 2	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.	

CODE

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

vector<pair<int, int>> adj[7];
vector<bool> v(7,false);
vector<int> dis(7, INT_MAX);

priority_queue<pair<int,int>, vector<pair<int,int>>,
greater<pair<int,int>>>pq;
void dji_algo(){

    pq.push({0,1});
    dis[1]=0;

    while(!pq.empty()){

        auto front = pq.top().second;
        v[front]=true;
        pq.pop();
```

```
        for(auto i:adj[front]){

            int wgt=i.second;
            int node = i.first;

           if(!v[node] && dis[front] + wgt < dis[node]) {
               dis[node]=dis[front]+wgt;
               pq.push({dis[node], node});
            }
        }


    }

}

void print() {
    for (int i = 1; i <=6; i++){
        cout<<"From Node 1"<<" to "<<i<<" is "<<dis[i]<<endl;
    }

}

int main()
{
    adj[1].push_back({2, 3});
    adj[2].push_back({1, 3});
    adj[1].push_back({4, 7});
    adj[4].push_back({1, 7});
    adj[2].push_back({5, 11});
    adj[5].push_back({2, 11});
    adj[3].push_back({1, 3});
    adj[1].push_back({3, 3});
    adj[3].push_back({2,7});
    adj[2].push_back({3,7});
    adj[5].push_back({1,3});
    adj[1].push_back({5,3});
    adj[5].push_back({4,5});
    adj[4].push_back({5,5});
    adj[5].push_back({6,7});
    adj[6].push_back({5,7});
    adj[6].push_back({4,1});
    adj[4].push_back({6,1});


    cout<<"Single Source Shortest Paths "<<endl;
```

```
    dji_algo();

    print();
    cout<<endl;

    #ifndef ONLINE_JUDGE
        cout<<"\nTime Elapsed : " << 1.0*clock() / CLOCKS_PER_SEC << " s\
n";
    #endif

    return 0;
}
```

OUTPUT

```
Current > ≡ output.txt
    1    Single Source Shortest Paths
    2    From Node 1 to 1 is 0
    3    From Node 1 to 2 is 3
    4    From Node 1 to 3 is 3
    5    From Node 1 to 4 is 7
    6    From Node 1 to 5 is 3
    7    From Node 1 to 6 is 8
    8
    9
   10    Time Elapsed : 0.005868 s
```

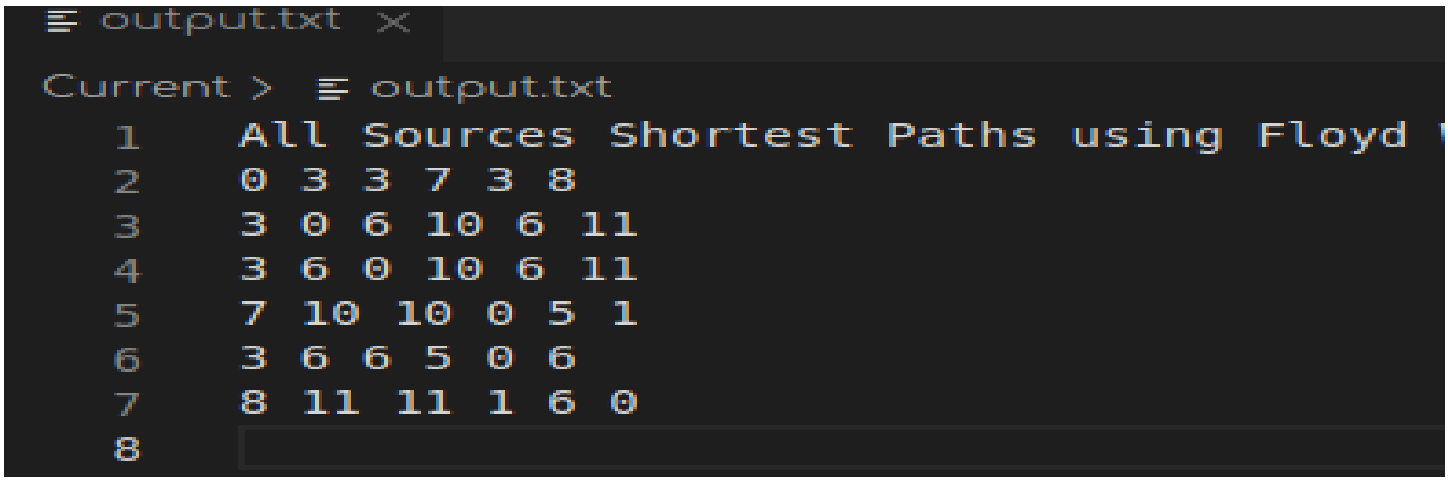
Question- 3 –a)	Implement All-Pairs Shortest Paths problem using Floyd's algorithm.	

CODE

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

vector<pair<int, int>> adj[7];
vector<vector<int>> adj_mat(6,vector<int>(6,INT_MAX));

void fw_algo(vector<vector<int>> &adj_mat){

    for (int k = 0; k < 6; k++)
   {
       for (int i = 0; i < 6; i++)
       {
          for (int j = 0; j < 6; j++)
          {
             if (adj_mat[i][k] + adj_mat[k][j] < adj_mat[i][j] && (adj_mat[k][j] != INT_MAX &&
adj_mat[i][k] != INT_MAX)){
                 adj_mat[i][j] = adj_mat[i][k] + adj_mat[k][j];
             }

          }
       }
   }


}

void print() {

  for (int i = 0; i < 6; i++) {
    for (int j = 0; j < 6; j++) {
      if (adj_mat[i][j] == INT_MAX){
         cout<<"INF"<<" ";
      }
      else{
        cout<<adj_mat[i][j]<<" ";
      }

    }
   cout<<endl;
  }

 }
```

```
int main()
{
   adj[1].push_back({2, 3});
   adj[2].push_back({1, 3});
   adj[1].push_back({4, 7});
   adj[4].push_back({1, 7});
   adj[2].push_back({5, 11});
   adj[5].push_back({2, 11});
   adj[3].push_back({1, 3});
   adj[1].push_back({3, 3});
   adj[3].push_back({2,7});
   adj[2].push_back({3,7});
   adj[5].push_back({1,3});
   adj[1].push_back({5,3});
   adj[5].push_back({4,5});
   adj[4].push_back({5,5});
   adj[5].push_back({6,7});
   adj[6].push_back({5,7});
   adj[6].push_back({4,1});
   adj[4].push_back({6,1});


   for(int i=1;i<=6;i++){
      adj_mat[i-1][i-1]=0;

      for(auto j:adj[i]){

         int sec = j.first;
         int val = j.second;

         adj_mat[i-1][sec-1] = val;
      }
   }

   cout<<"All Sources Shortest Paths using Floyd Warshall "<<endl;

   fw_algo(adj_mat);

   print();

   return 0;
}
```

OUTPUT



Question – 3-b)	Implement Travelling Sales Person problem using Dynamic programming.	

CODE

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

vector<pair<int, int>> adj[5];
vector<vector<int>> adj_mat(4,vector<int>(4,INT_MAX));
int dp[20][10];

int n = 4;

int compute(int ma_in,int curr,int &ma){

   if(ma_in==ma){
      return adj_mat[curr][0];
   }
```

```
    if(dp[ma_in][curr]!=-1){
          return dp[ma_in][curr];
       }

    int ans = INT_MAX;

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

             if((ma_in&(1<<i))==0) {
          int temp = adj_mat[curr][i] + compute(ma_in|(1<<i), i,ma);
                   ans = min(ans, temp);

             }

       }

       dp[ma_in][curr] = ans;
       return dp[ma_in][curr];


}

int main()
{
    adj[1].push_back({2, 3});
    adj[2].push_back({1, 3});
    adj[1].push_back({4, 7});
    adj[4].push_back({1, 7});
    adj[3].push_back({1, 3});
    adj[1].push_back({3, 3});
    adj[3].push_back({2,7});
    adj[2].push_back({3,7});
    adj[2].push_back({4,5});
    adj[4].push_back({2,5});
    adj[3].push_back({4,6});
    adj[4].push_back({3,6});

    for(int i=1;i<=4;i++){
       adj_mat[i-1][i-1]=0;

       for(auto j:adj[i]){

          int sec = j.first;
          int val = j.second;
          adj_mat[i-1][sec-1] = val;
       }
```

```
    }


    cout<<"Minimum Weight Sum for a Travelling Salesman is"<<endl;

    for(int i=0;i<(1<<n);i++){
        for(int j=0;j<n;j++){
            dp[i][j] = -1;
        }
    }

    int ma = (1<<n) - 1;

    cout<<compute(1,0,ma);

    return 0;
}
```

OUTPUT