```
/*Name - Amod Dhopavkar
Roll No - 33304

Bellman Ford using DP
*/
```

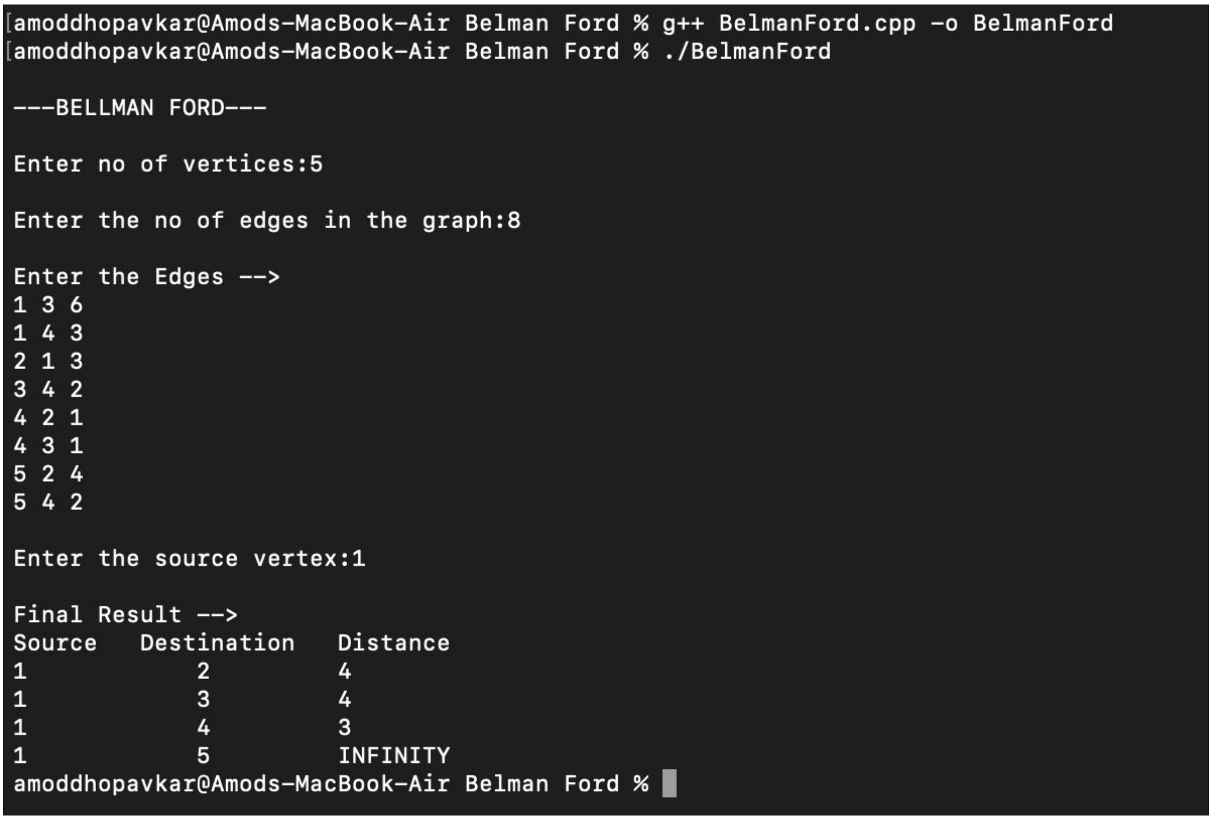
Code →

```
#include<iostream>
using namespace std;

typedef struct graph {
  int s;
  int d;
  int w;
}graph;

int checkvalidation(struct graph *arr,int size,int s,int d) {
    for(int i=0;i<size;i++) {
        //cout<<arr[i][0]<<" "<<arr[i][1]<<"      "<<s<<" "<<d<<endl;
        if((arr[i].s==s) && (arr[i].d==d) )
        return 0;
    }
    return 1;
}

int checknegativecyle(struct graph *gr,int *arr,int t) {
    int flag=0,temp;
    for(int i=0;i<t;i++) {
        if(arr[gr[i].s-1]!=INT_MAX) {
            temp=arr[gr[i].s-1]+gr[i].w;
                if(arr[gr[i].d-1]>temp) {
                    arr[gr[i].d-1]=temp;flag=1;
                }
        }
    }
       // cout<<"flag="<<flag<<endl;
    if(flag==1) {
        cout<<"Negative Cycle Exists"<<endl;
        return 1;
    }
    return 0;
}

int main() {
    int n=0,s,t=0;
```

```
cout<<"\n---BELLMAN FORD---"<<endl;
cout<<"\nEnter no of vertices:";
cin>>n;
while (n < 3) {
    cout<<"\nEnter a valid no vertices:";
}

int *arr=(int*)malloc((n)*sizeof(int));
for(int i=0; i<n; i++) {
    arr[i]=INT_MAX;
}

while (t < n-1) {
  cout<<"\nEnter the no of edges in the graph:";
  cin>>t;
  if (t < n-1)
  cout<<"\nEnter a valid no of edges:";
}

graph *gr = (graph*)malloc(t*sizeof(graph));
cout<<"\nEnter the Edges -->"<<endl;
for (int i=0; i<t; i++)
{
    int s,d,w;
    cin>>s>>d>>w;
    if(s!=d && (s>n || d>n || s<1 || d<1)) {
        cout<<"\nRe-Enter the Edge:";
        i--;
    }
    else {
        if(checkvalidation(gr,i,s,d)) {
            gr[i].s=s;gr[i].d=d;gr[i].w=w;
        }
        else {
            cout<<"\nRe-Enter the Edge:";
            i--;
        }
    }
}
s = -1;
while (s<1 || s >n) {
    cout<<"\nEnter the source vertex:";
    cin>>s;
}
arr[s-1]=0;
long int temp;
```

```
    int flag=0,f=0;
    for (int j=1; j<n; j++) {
        flag=0;
        for (int i=0; i<t; i++) {
                if(arr[gr[i].s-1]!=INT_MAX) {
                temp=arr[gr[i].s-1]+gr[i].w;
                    if(arr[gr[i].d-1]>temp) {
                        arr[gr[i].d-1]=temp;flag=1;
                    }
                }
                else
                    f=1;
        }
        if(flag==0)
            break;
    }
   if (f != 1)
   if (checknegativecyle(gr,arr,t)) {
        free(arr);
        free(gr);
        return 0;
    }

    cout<<"\nFinal Result -->"<<endl;
    cout<<"Source   Destination   Distance"<<endl;
    for (int i=0; i<n; i++) {
        if(i != s-1) {
            if(arr[i] == INT_MAX)
                cout<<s<<"            "<<i+1<<"         INFINITY";
            else
                cout<<s<<"            "<<i+1<<"         "<<arr[i];
            cout<<endl;
        }
    }
}
```

Output →



```
[amoddhopavkar@Amods-MacBook-Air Belman Ford % g++ BelmanFord.cpp -o BelmanFord
[amoddhopavkar@Amods-MacBook-Air Belman Ford % ./BelmanFord

---BELLMAN FORD---

Enter no of vertices:5

Enter the no of edges in the graph:8

Enter the Edges -->
1 3 6
1 4 3
2 1 3
3 4 2
4 2 1
4 3 1
5 2 4
5 4 2

Enter the source vertex:1

Final Result -->
Source   Destination   Distance
1            2         4
1            3         4
1            4         3
1            5         INFINITY
amoddhopavkar@Amods-MacBook-Air Belman Ford % 
```