INPUT:

```
#include<iostream>
using namespace std;
class Flyod_Warshall{
   int A[100][100];
   int n;
public:
   Flyod_Warshall()
   {
      cout<<"\n\tWelcome to All-pairs Shortest Path of a Graph";
      cout<<"\nEnter number of vertices: ";
      cin>>n;
      for(int i=0; i<n; i++)
         A[i][i]=0;
   }
   void create();
   void display();
   void algo();
};
void Flyod_Warshall::create()
{
   char ans='y';
   int v1,v2, cost;
   cout<<"\nEnter Directed edges of graph below:\n";
   do{
      cout<<"\nEnter edge pair in (v1-v2) form: ";
      cin>>v1>>v2;
      cout<<"\nEnter cost of edge: ";
      cin>>cost;
      A[v1][v2]=cost;
      cout<<"\nContinue?(Y/N): ";
      cin>>ans;
   }while(ans=='y' || ans=='Y');
}

void Flyod_Warshall::display()
{
   for(int i=0; i<n; i++)
   {
      cout<<endl;
      for(int j=0; j<n; j++)
         cout<<A[i][j]<<"  ";
   }
}

void Flyod_Warshall::algo()
{
   cout<<"\nAdjacency Matrix of graph: ";
   this->display();
   for(int k=0; k<n; k++)
   {
```

```
      for(int i=0; i<n; i++)
      {
         for(int j=0; j<n; j++)
         {
            if((i!=j) && (A[i][j]!=0) && (A[i][j]>(A[i][k]+A[k][j])))
               A[i][j]=A[i][k]+A[k][j];
         }
      }
      if(k!=n-1)
      {
         cout<<"\nMatrix after "<<k+1<<" iteration: ";
         this->display();
      }
   }
   cout<<"\nFinal Matrix with shortest path b/w all pairs after last iteration is: ";
   this->display();
}

int main()
{
   Flyod_Warshall o;
   o.create();
   o.algo();
   cout<<"\n\tProgram Ends!";
   return 0;
}
```

OUTPUT:

```
      Welcome to All-pairs Shortest Path of a Graph
Enter number of vertices: 3

Enter Directed edges of graph below:

Enter edge pair in (v1-v2) form: 0
1

Enter cost of edge: 5

Continue?(Y/N): y

Enter edge pair in (v1-v2) form: 1
0

Enter cost of edge: 4

Continue?(Y/N): y

Enter edge pair in (v1-v2) form: 1
2
```

```
Enter cost of edge: 8

Continue?(Y/N): y

Enter edge pair in (v1-v2) form: 2
1

Enter cost of edge: 1

Continue?(Y/N): y

Enter edge pair in (v1-v2) form: 0
2

Enter cost of edge: 2

Continue?(Y/N): y

Enter edge pair in (v1-v2) form: 2
0

Enter cost of edge: 10

Continue?(Y/N): n

Adjacency Matrix of graph:
0  5  2
4  0  8
10  1  0
Matrix after 1 iteration:
0  5  2
4  0  6
10  1  0
Matrix after 2 iteration:
0  5  2
4  0  6
5  1  0
Final Matrix with shortest path b/w all pairs after last iteration is:
0  3  2
4  0  6
5  1  0
      Program Ends!
```