California State University Fullerton

Professor: Doina Bein

Project 1 Report: Savvy Traveler
CPSC 535 -01 (14069)

Submission by:

Fly with algo group

Name	Email	Position
NIDHI SHAH	nidhishah989@csu.fullerton.edu	Team member
MERIN JOSEPH	merin.joseph@csu.fullerton.edu	Team member
APEKSHA SHAH	apeksha@csu.fullerton.edu	Team member

Table of Contents:

1. Summary of Project - Savvy Traveler

1.1 Problem Statement

This problem is air travel based. From many available flights, users would like to choose one flight from one point (city) to another (city). The problem is to calculate what route will maximize the probability to arrive on time between two cities. In addition, the problem is to find the city that is the most reliable travel destination.

1.2 Assumption factors (given with problem)

- We know what probability each flight is on time.
- Between two cities (nodes), there is the same probability of travel from one to another.

1.3 Input and Output Summary

Input:

- graph (G= (V,E))with cities and probability of travel between two cities.
- Source city from where user would like to travel
- Destination city to where user would like to travel

Output:

- The highest probability path with highest probability computation of on-time arrival.
- The city which is the most reliable travel destination among given cities.

1.4 Probability Calculation Technique

- If the path between two cities has a single edge then that edge probability will be the travel probability between those two cities.
- If the path between two cities has multiple edges then the multiplication of all edge probabilities will be the travel probability between those two cities.
- If alternating paths are available between two cities, select the path with highest probability.
- The city reliability measurement will be the sum of all probabilities of on-time arrivals from each other city to that city.

2. Pseudocode of the Algorithm

2.1 Design Overview

The user will provide the graph with vertices(cities) and edges (travel probabilities) and also provide the source and destination cities to find the highest probability of on-time arrival path. The algorithm will use the dijkstra algorithm technique with such data structure that will provide the highest probability city in less time complexity. First, the function will traverse all cities (V) from the graph and set the storage data structure to store all probabilities from that city (V) to other cities. The probability will be calculated with given probability techniques with finding the highest probability edge(E) between cities. After finding all cities on-time arrival probabilities from the source city, the function will calculate the reliability of that source city as the sum of those probabilities. If the traverse source vertex(city) is the user's source city, then it will find the path with highest probability of on-time arrival value to the user's destination city. After the main traverse completes for all cities, the function will find the most reliable city with highest reliability. At the end, it will show all outputs.

2.2 General Pseudocode

```
    function flyontime(Graph(V,<N,EPrb>), source,destination):
    {
        //map to store the new prob for connecting city(U) and store 'S' (source city -the reason of
         changing U(Prob)
         Cityticityprob map < 'U',{Prob,'S'}> // initialize with Prob = -1 (infinity) and 'S' as '#'
         nextmaxprobcity  Sorted set by pair <Prob,'V'> //descending sort by probability : max first
         Finalreliability Sorted set by pair <RE,'Node'> //descending sort by reliability: max first
         for each vertex V in Graph(V,<N,EPrb>)
         {
          Cityticityprob [V] = 0   // probability of source node is 0
          currentcity= V
          Set Visited.insert(V)
          while (visited.size() < Graph(V,E).size())
          {
             for each neighbor city N of V
             {
               If N is not visited:
                 If Prob(V) is 0:
                    product= EPrb(N) from graph
                 Else
                  product=Prob(currentcity) from Cityticityprob map * EPrb(N from currentcity) from graph
                  IF product > Prob(N) from Cityticityprob map
                           {
                             Prob(N) from Cityticityprob map = product
                             nextmaxprobcity.insert(Prob(N),N)
                           }
             } //inner for loop end
             Currentcity = nextmaxprobcity-> first element // max prob cause descending sorted set
             visited.insert(Currentcity )
             nextmaxprobcity.erase(first element)
           }//while loop end
          for each U in  Cityticityprob map < 'U',{Prob,'S'}>
          {
           Reliability(V) = Reliability(V) + Prob(U)
          }
           Finalreliability Sorted set.insert( Reliability(V), V)
         If V == source{
                  Check = destination
                  Path =destination
                  Finalhighprob = Cityticityprob(check).Prob
                 while(until check == source)
                 {
                     Path =  Cityticityprob(check).'S' + Path
```

```
                }
            }
            }// end of main for loop
        //show all outputs
        Output<< Finalhighprob
        Output << Path
        Output << Finalreliability -> first element. Node // max reliability cause descending sorted set
} //end of algorithm
```

2.3 Programming Language

This Algorithm is written with consideration that the programming language is c++ for project code.

2.4 Pseudocode with C++ (detail)

```
void flyontime(graph(V,E), source, destination)
{
   //output variables declaration
   double  highprobpathprob=0.0;
   string highprobpath ="";

   //set-> to store the reliability from other cities
   set < pair < double, char >, greater <pair < double, char >> > cityreliability;
   // Initialize an iterator to iterate through the set for finding the most reliable city with
    maximum probability
   set < pair < double, char > > ::iterator maxcity;

   //iteration declaration for graph(V,E)
   map<char, map<char, double>>::iterator it;
   //iteration declaration for inside map of graph-> (E,prob)
   map<char, double>::iterator inside;

   // traverse the given graph for each city (V)
   for (it = graph.begin(); it != graph.end(); it++)
     {
       //set to store nextmaxprobablecity
        set < pair < double, char >, greater <pair < double, char >> >   nextmaxprobcity;
       //iterator 'next' to iterate through the set netmaxprobcity
       set < pair < double, char > > ::iterator next;
       //declare map to store probability of each city from source city(V), 'V'--> {Prob,sourceV}
      //-1 for infinity and # to store the source city that will change the key city probability
      map<char, pair<double, char>> citytocitiesprobability=
                                                          { {'A',{-1,'#'}},
                                                             {'B',{-1,'#'}},
                                                             {'C',{-1,'#'}},
                                                             {'D',{-1,'#'}},
                                                             {'E',{-1,'#'}},
                                                                        {'F',{-1,'#'}},
                                                                        {'G',{-1,'#'}},
                                                                        {'H',{-1,'#'}} };


      //iterator final for citytocitiesprobability map
      map<char, pair<double, char>> ::iterator final;
      char currentvisitcity;
      double probproduct;  //probability product for connecting cities from visited city
      double totalprobforcity = 0.0; //final sum output variable declaration
      //set to store the visited city node
      set <char> visited;
      //iterator for visited set
      set<char>::iterator ps;
     citytocitiesprobability.at(sourceV).Prob=0;
     visited.insert(sourceV);
     currentvisitcity=SourceV;
    while (visited.size() < graph.size())
    {
        // loop through all connected nodes(U) for the it->first node using the inner map's
         iterator inside.
       for (inside = graph.at(currentvisitcity).begin(); inside!= graph.at(currentvisitcity).end(); inside++)
        {
          //check the connected city(U) is visited or not
          ps = visited.find(inside->first);
          if ps=visited.end()   //production calculation for all connecting nodes which are not visited
           {
               If (case 1: if city(U) not visited and the currentcity prob is 0(means source city(V)))
                {
                  //initialize value of product to the first edge weight
                   product = inside->second;
                }
               Else (case 2: if city(U) not visited and currentcity prob is not 0:)
                 {
                    //calculate the production probability for multiedge with currentcity probability
                    product = citytocitiesprobability.at(currentvisitcity).first *
                                    graph.at(currentvisitcity).at(inside->first);
                 }
                //check the production probability is higher than U(current probability)
               if (product > citytocitiesprobability.at(inside->first).first)
                {
                       citytocitiesprobability.at(inside->first).first=product;
                      // change the source city for this U(key city for future find path option)
                      citytocitiesprobability.at(inside->first).second = currentvisitcity;

                     //insert new pair of (U,Prob) in the nextprob sorted set
```

```
                        nextmaxprobcity.insert({ inside->second, inside->first });
                    }
                } //end if inside for loop

                //make next high prob city to current city(visiting node)
                 next = nextmaxprobcity.begin();
                 visited.insert(next->second);
                 currentvisitcity = next->second;
                 //erase the visited city from the nextmaxprobcity set.
                  nextmaxprobcity.erase(next);
            }//end of while loop

            //find the total reliability for the source city(V)
           for (final = citytocitiesprobability.begin(); final != citytocitiesprobability.end(); final++)
            {
            totalprobforcity = totalprobforcity + citytocitiesprobability.at(final->first).first;
             }
            //insert the total reliabiity for source city
            cityreliability.insert({ totalprobforcity,it->first });

          //find the high prob path and probability if the source (V) is given user's source
          char check=destination;
          char check = destination;
          highprobpathprob = citytocitiesprobability.at(destination).first;
          highprobpath = destination;
            //while loop until the source city is reached
            while (check != source)
            {
                    highprobpath = citytocitiesprobability.at(check).second + highprobpath;
                    check = citytocitiesprobability.at(check).second;
            }


      }//end of uper for loop
      //show the output
     cout<< highprobpathprob ;
     Cout << highprobpath ;
     //find the high reliable city
     maxcity = cityreliability.begin();
    Cout<<maxcity;
}//end of function

int main()
{
    // to create graph
    scanf>> Ask user for first node
    scanf>>Ask users to provide all connecting nodes and its probability
    Create Graph in map<char, map<char, double>> format
    scanf>> source city;
    scanf >> destination city;
    flyontime(map, source,destination)

     Return 0;
}
```

2.5 Algorithm Complexity

The time complexity for the algorithm designed is **O(V^2 + EVlogV)**, where V is the total number of vertices/nodes in the graph and E is the total number of edges in the graph.

3. Description on how to run the code

Code file name: flyontime.cpp
From terminal to run code: `g++ -o <name-you-want-to-give> flyontime.cpp`
Run `./name-you-give`

NOTE: We have two ways to input the graph, source and destination.

1. **Manually from main function: Direct graph in the form of map and source and destination.**
    - **Comment line of codes: from line 167 to line 204.**
    - **Uncomment line of codes: from line 210 to line 255**
2. **Ask users to enter the nodes and edges to create the graph and source and destination.**

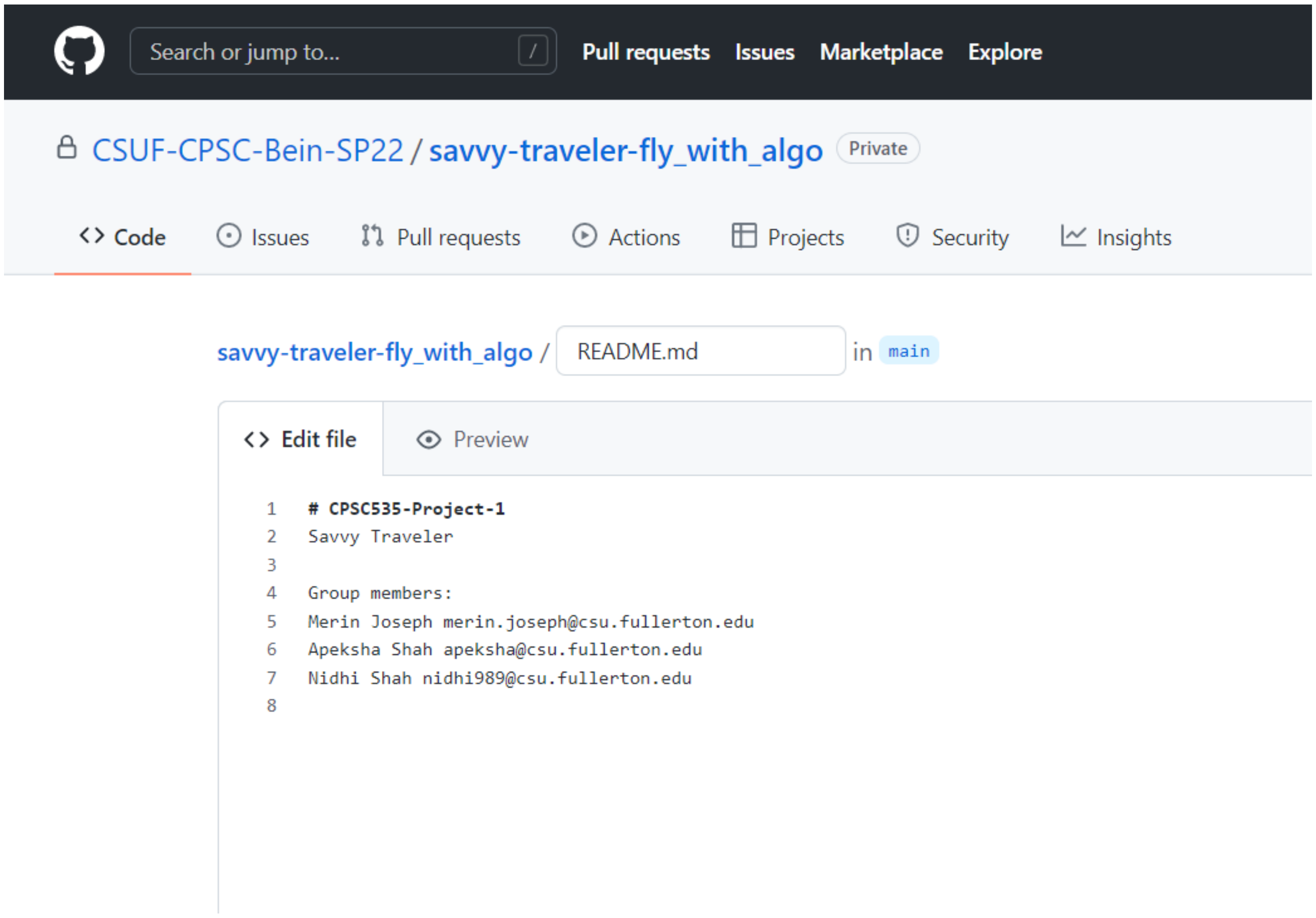
Below is the explanation of how users can input the graph.
The algorithm needs the graph, source node and destination node to function. The user can input any graph of his choice, provided he specifies the nodes and their connections along with the edge weights properly. The proper format for inputting the graph is as follows:
...............ENTER YOUR GRAPH................

1. Enter the node:
   ← enter the first node in your graph →
2. Enter the connected NODE to ←node→ and PROBABILITY:
   ← enter the node connected to the above given node and the probability of their edge →
3. Press 1 to add connected nodes or press 2 to end:
   ← if the user press 1, the user can keep on adding the nodes connected to the other node. If the user presses 1, then the second step of reading the connected node and weight repeats. If there are no more connected nodes to be added then, the user needs to press 2. On pressing 2, the user exits to the next step of adding more nodes to the graph. →
4. Press 0 to add a new node or press 2 to end:
   ← The user need to press 0 inorder to add more nodes to the graph. On pressing 1, steps 1 to 3 repeat, so that the user can add its corresponding connected nodes too. On pressing 2, user cannot input any more nodes or edges to the graph. →

5. Enter the SOURCE node:
   ← The user can now input the source node from which he needs to travel from →
6. Enter the DESTINATION node:
   ← The user can enter the destination city where he needs to travel to →
7. The results are then displayed showing the highest probability and its path of on-time arrival from given source to destination. Also, it shows the most reliable travel destination city as output.

4. Screenshots

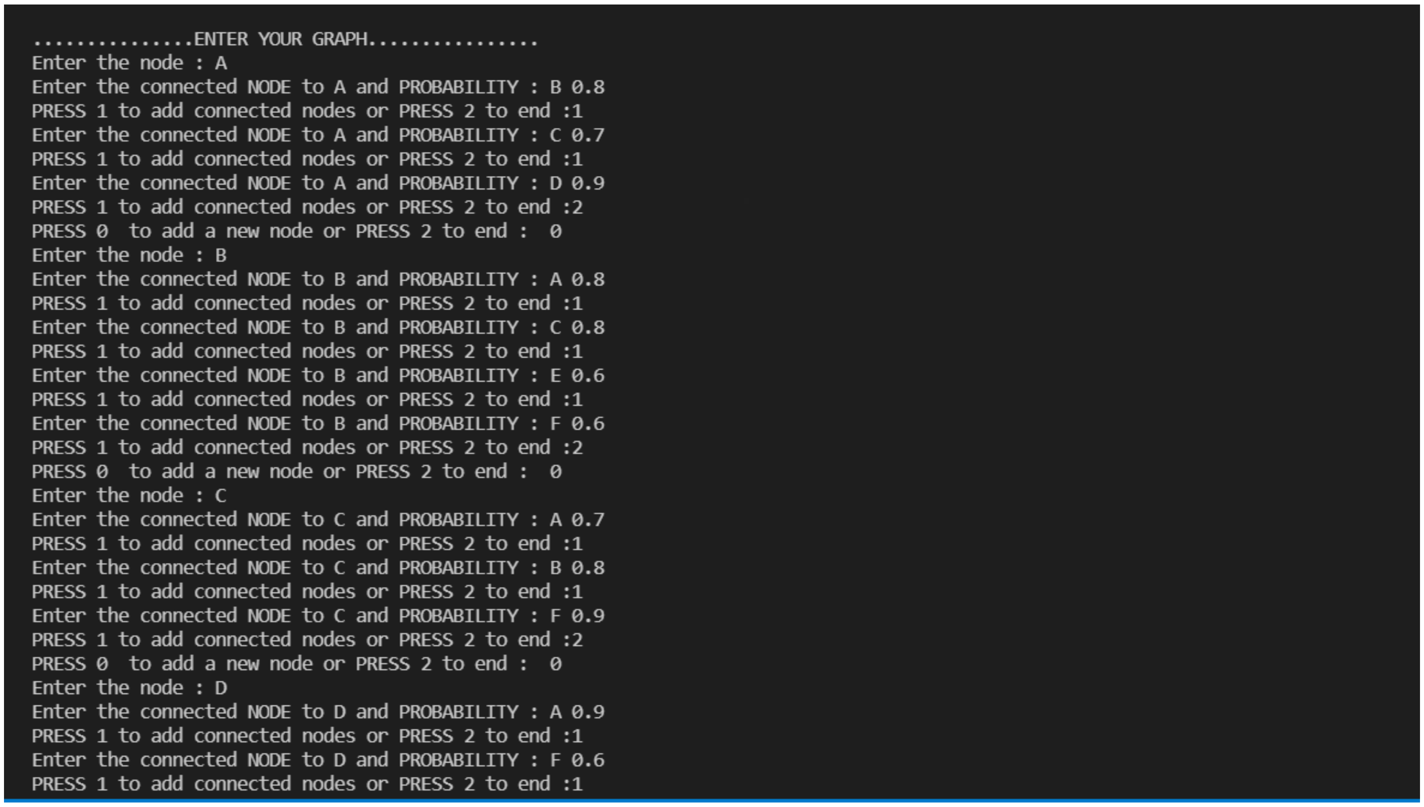
4.1 Group member screenshot



```
# CPSC535-Project-1
Savvy Traveler

Group members:
Merin Joseph merin.joseph@csu.fullerton.edu
Apeksha Shah apeksha@csu.fullerton.edu
Nidhi Shah nidhi989@csu.fullerton.edu
```

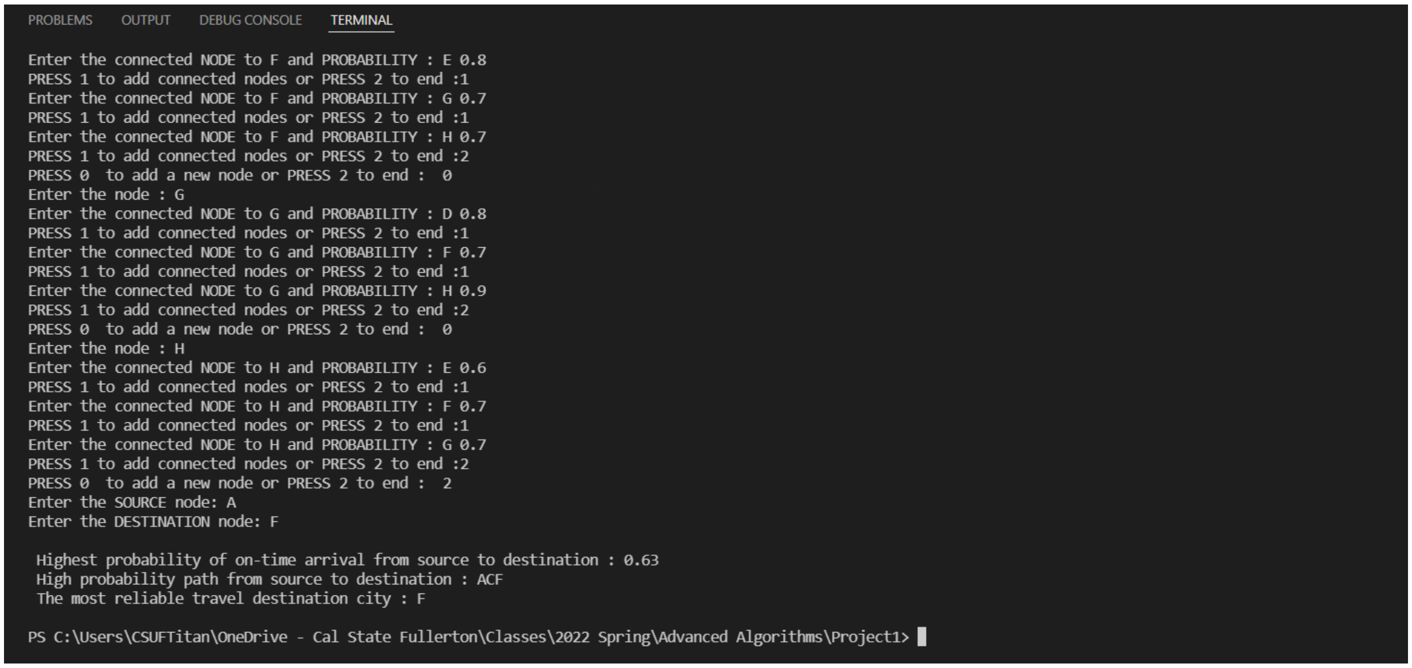
4.2 Example 1 Screenshot (With user input description)

- This screenshot shows how users need to enter the inputs for graph, source and destination.
- And run Example 1 and show output.



```
...............ENTER YOUR GRAPH...............
Enter the node : A
Enter the connected NODE to A and PROBABILITY : B 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to A and PROBABILITY : C 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to A and PROBABILITY : D 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : B
Enter the connected NODE to B and PROBABILITY : A 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to B and PROBABILITY : C 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to B and PROBABILITY : E 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to B and PROBABILITY : F 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : C
Enter the connected NODE to C and PROBABILITY : A 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to C and PROBABILITY : B 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to C and PROBABILITY : F 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : D
Enter the connected NODE to D and PROBABILITY : A 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to D and PROBABILITY : F 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
```

-

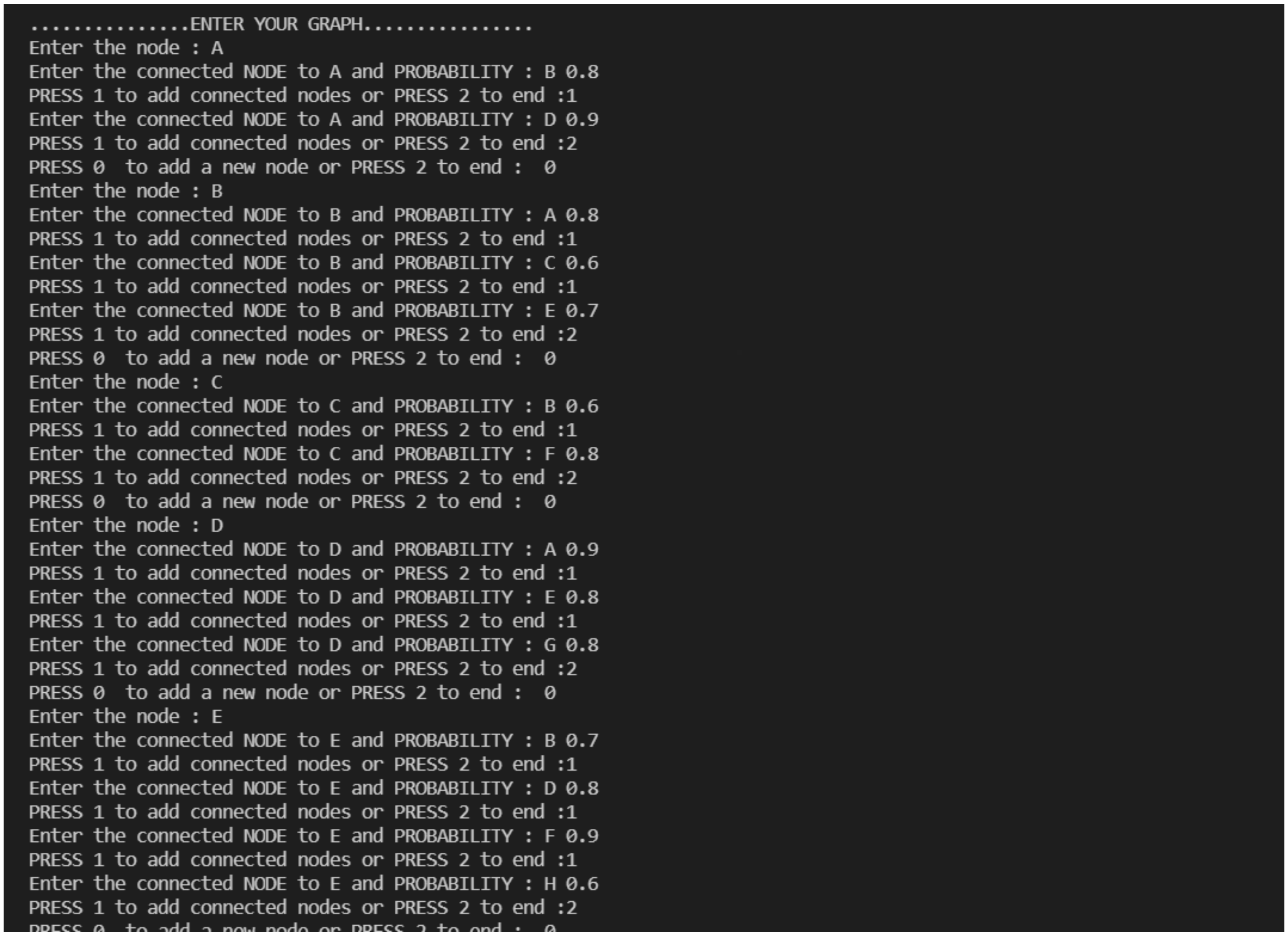


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Enter the connected NODE to F and PROBABILITY : E 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to F and PROBABILITY : G 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to F and PROBABILITY : H 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : G
Enter the connected NODE to G and PROBABILITY : D 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to G and PROBABILITY : F 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to G and PROBABILITY : H 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : H
Enter the connected NODE to H and PROBABILITY : E 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to H and PROBABILITY : F 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to H and PROBABILITY : G 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  2
Enter the SOURCE node: A
Enter the DESTINATION node: F

 Highest probability of on-time arrival from source to destination : 0.63
 High probability path from source to destination : ACF
 The most reliable travel destination city : F

PS C:\Users\CSUFTitan\OneDrive - Cal State Fullerton\Classes\2022 Spring\Advanced Algorithms\Project1>
```

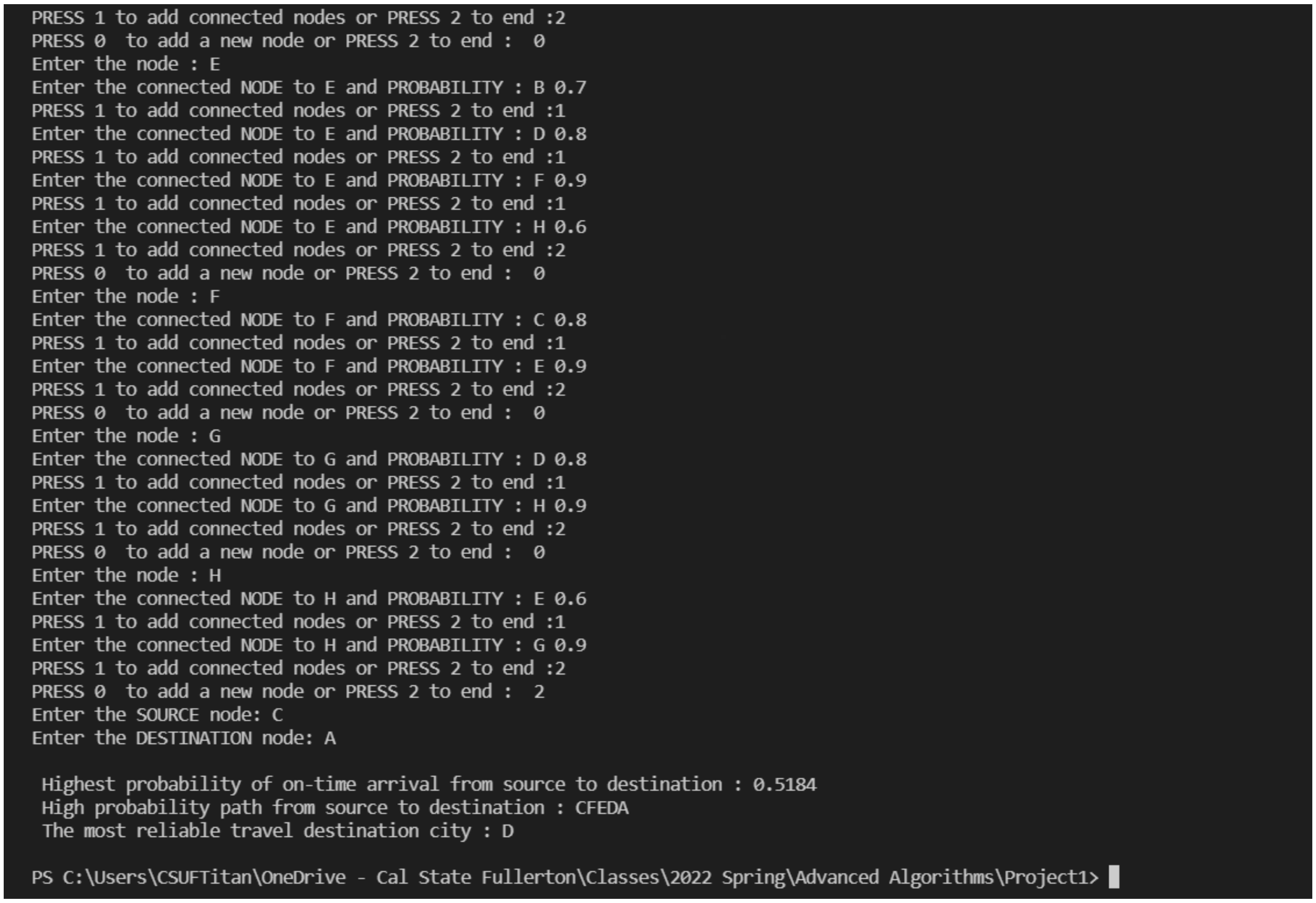
4.3 Example 2 Screenshot (With user input description)

- This screenshot shows how users need to enter the inputs for graph, source and destination.
- And run Example 1 and show output.



```
...............ENTER YOUR GRAPH...............
Enter the node : A
Enter the connected NODE to A and PROBABILITY : B 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to A and PROBABILITY : D 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : B
Enter the connected NODE to B and PROBABILITY : A 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to B and PROBABILITY : C 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to B and PROBABILITY : E 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : C
Enter the connected NODE to C and PROBABILITY : B 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to C and PROBABILITY : F 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : D
Enter the connected NODE to D and PROBABILITY : A 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to D and PROBABILITY : E 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to D and PROBABILITY : G 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : E
Enter the connected NODE to E and PROBABILITY : B 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : D 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : F 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : H 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
```

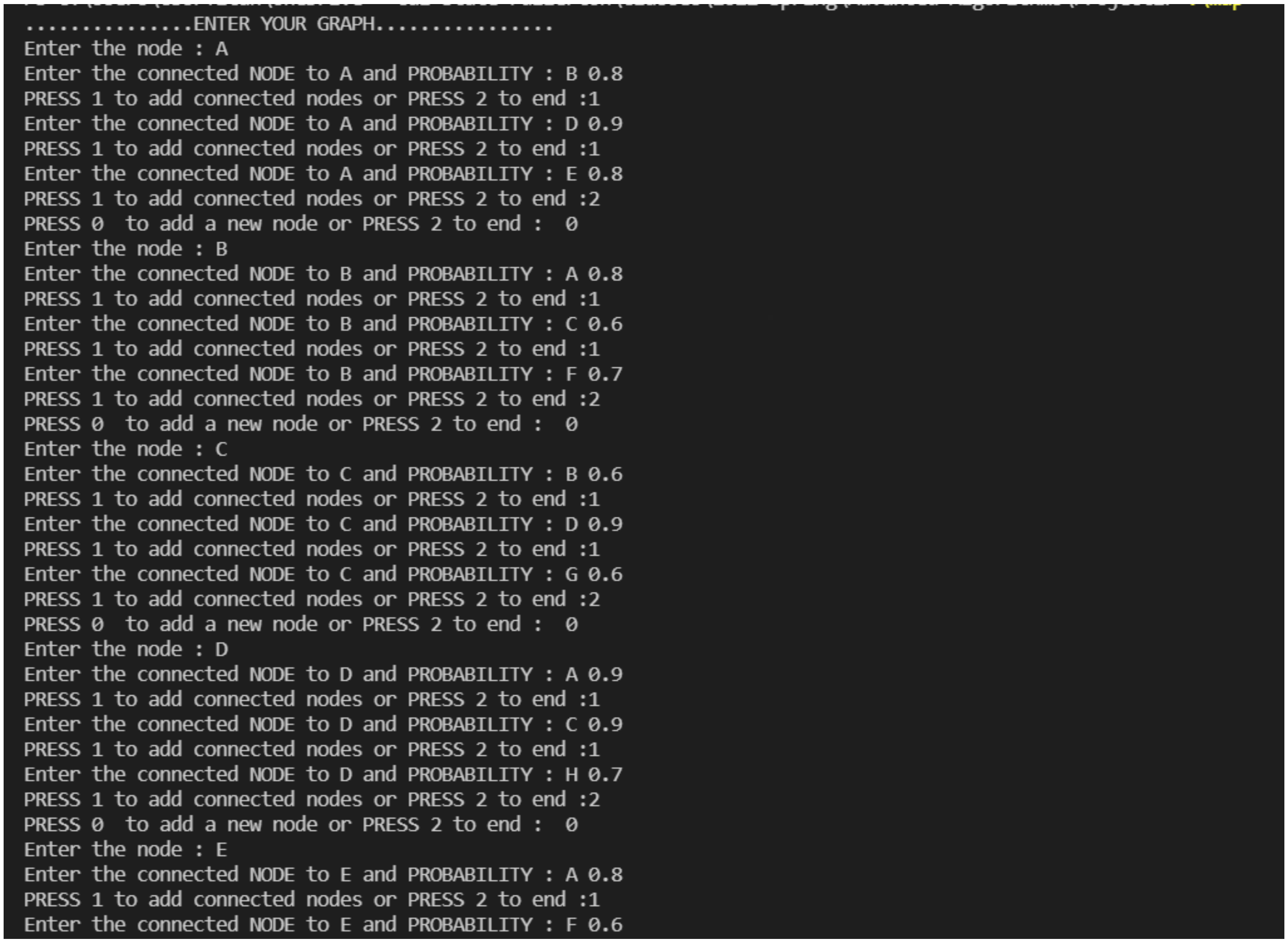
-

- 

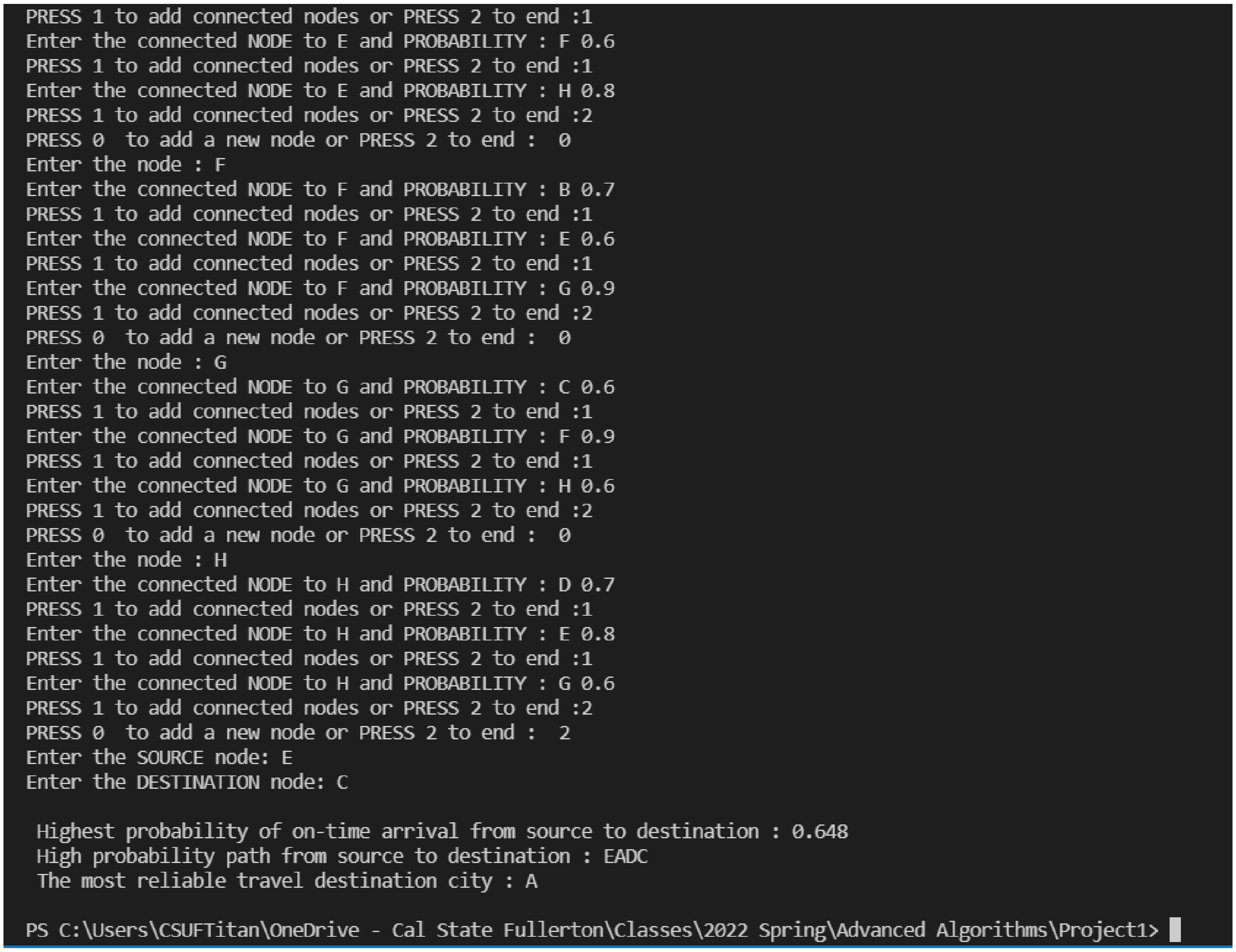
```
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : E
Enter the connected NODE to E and PROBABILITY : B 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : D 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : F 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : H 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : F
Enter the connected NODE to F and PROBABILITY : C 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to F and PROBABILITY : E 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : G
Enter the connected NODE to G and PROBABILITY : D 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to G and PROBABILITY : H 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : H
Enter the connected NODE to H and PROBABILITY : E 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to H and PROBABILITY : G 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  2
Enter the SOURCE node: C
Enter the DESTINATION node: A

 Highest probability of on-time arrival from source to destination : 0.5184
 High probability path from source to destination : CFEDA
 The most reliable travel destination city : D

PS C:\Users\CSUFTitan\OneDrive - Cal State Fullerton\Classes\2022 Spring\Advanced Algorithms\Project1>
```

4.4 Example 3 Screenshot (With user input description)

- This screenshot shows how users need to enter the inputs for graph, source and destination.
- And run Example 1 and show output.



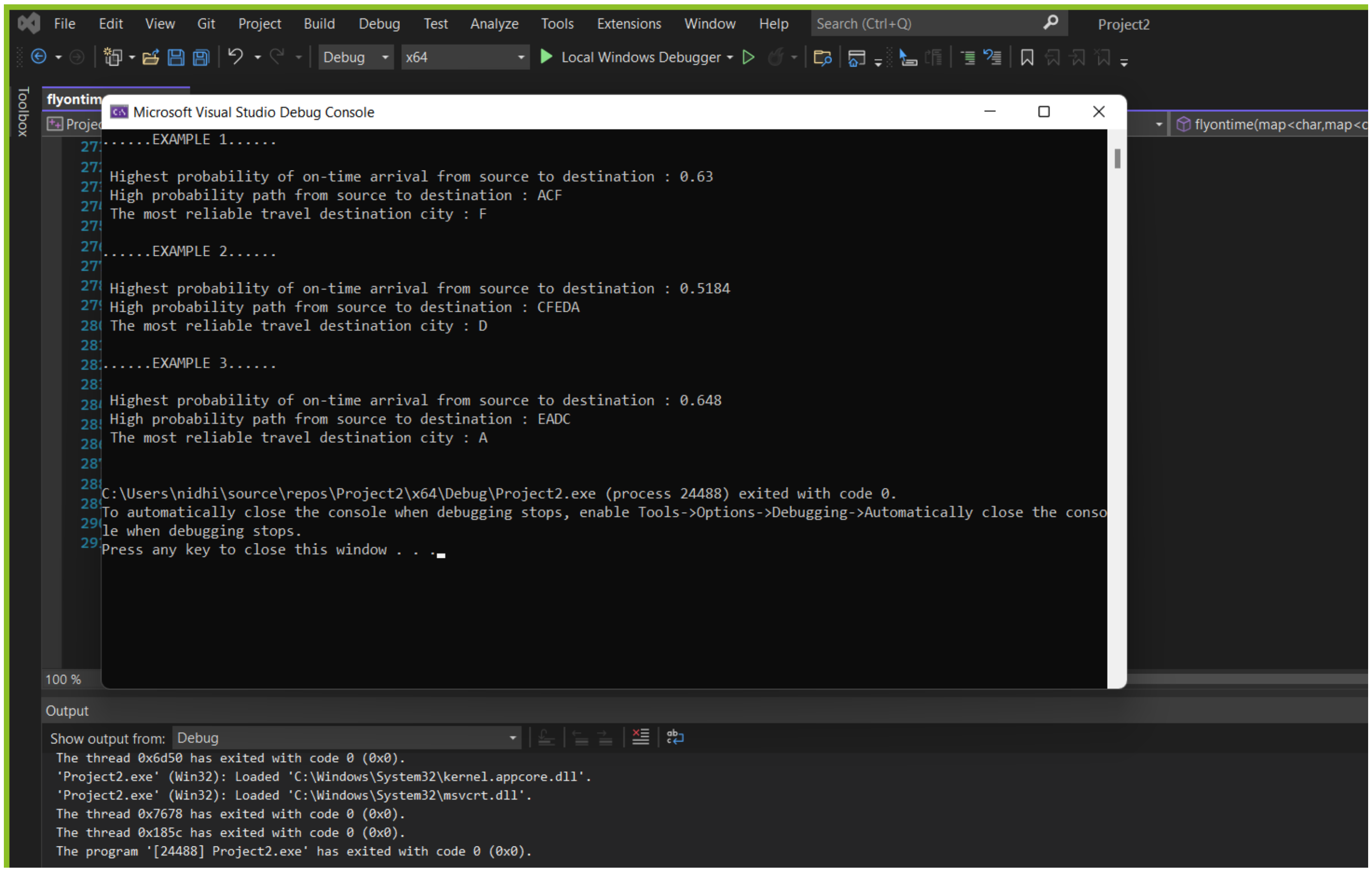
```
...............ENTER YOUR GRAPH...............
Enter the node : A
Enter the connected NODE to A and PROBABILITY : B 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to A and PROBABILITY : D 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to A and PROBABILITY : E 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : B
Enter the connected NODE to B and PROBABILITY : A 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to B and PROBABILITY : C 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to B and PROBABILITY : F 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : C
Enter the connected NODE to C and PROBABILITY : B 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to C and PROBABILITY : D 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to C and PROBABILITY : G 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : D
Enter the connected NODE to D and PROBABILITY : A 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to D and PROBABILITY : C 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to D and PROBABILITY : H 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : E
Enter the connected NODE to E and PROBABILITY : A 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : F 0.6
```

- 

```
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : F 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to E and PROBABILITY : H 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : F
Enter the connected NODE to F and PROBABILITY : B 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to F and PROBABILITY : E 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to F and PROBABILITY : G 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : G
Enter the connected NODE to G and PROBABILITY : C 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to G and PROBABILITY : F 0.9
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to G and PROBABILITY : H 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  0
Enter the node : H
Enter the connected NODE to H and PROBABILITY : D 0.7
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to H and PROBABILITY : E 0.8
PRESS 1 to add connected nodes or PRESS 2 to end :1
Enter the connected NODE to H and PROBABILITY : G 0.6
PRESS 1 to add connected nodes or PRESS 2 to end :2
PRESS 0  to add a new node or PRESS 2 to end :  2
Enter the SOURCE node: E
Enter the DESTINATION node: C

 Highest probability of on-time arrival from source to destination : 0.648
 High probability path from source to destination : EADC
 The most reliable travel destination city : A

PS C:\Users\CSUFTitan\OneDrive - Cal State Fullerton\Classes\2022 Spring\Advanced Algorithms\Project1>
```

4.5 Alternate Way of graph Input (Manually input in program as map)

- This screenshot shows the output of all examples which graph and source and destination given manually from the main function.



```
......EXAMPLE 1......

 Highest probability of on-time arrival from source to destination : 0.63
 High probability path from source to destination : ACF
 The most reliable travel destination city : F

......EXAMPLE 2......

 Highest probability of on-time arrival from source to destination : 0.5184
 High probability path from source to destination : CFEDA
 The most reliable travel destination city : D

......EXAMPLE 3......

 Highest probability of on-time arrival from source to destination : 0.648
 High probability path from source to destination : EADC
 The most reliable travel destination city : A

C:\Users\nidhi\source\repos\Project2\x64\Debug\Project2.exe (process 24488) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
The thread 0x6d50 has exited with code 0 (0x0).
'Project2.exe' (Win32): Loaded 'C:\Windows\System32\kernel.appcore.dll'.
'Project2.exe' (Win32): Loaded 'C:\Windows\System32\msvcrt.dll'.
The thread 0x7678 has exited with code 0 (0x0).
The thread 0x185c has exited with code 0 (0x0).
The program '[24488] Project2.exe' has exited with code 0 (0x0).
```