**CAB BOOKING SERVICES**

**A**

**Mini Project Report**

**Submitted for the course**

**Design and Analysis of Algorithms Lab**

**Semester-IV, CSE-B**

**By**

**K.ANISHA 1602-18-733-068**

**G.MEGHANA 1602-18-733-086**



**Department of Computer Science and Engineering**

**VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)**

**9-5-81, Ibrahimbagh, Hyderabad-31**

**2019-20**

**ACKNOWLEDGEMENT**

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Mrs. V Sireesha,** Department of Computer Science and Engineering. Whose guidance was unforgettable and innovative ideas as well as his constructive suggestions has made the presentation of my thesis a grand success.

We are thankful to **Dr. T. Adilakshmi,**   Head of Department (CSE), **Vasavi College of Engineering** for their help during our course work.

Finally at last but not least express our heart full thanks to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete my project work successively.

**TABLE OF CONTENTS**

**S.no. Topic name Pg.no.**

**1. Project Abstract 5**

**2. Description 6-7**

**3. Code 8-15**

**4. Output 16-17**

**5. Constraints &Test Cases 18-21**

**6. Conclusion 22**

**7. References 23**

**LIST OF FIGURES**

**1. Structure of a graph…………………………………………….fig1-6**

**2. Shortest path algorithm………………………………………….fig2-7**

**PROJECT -ABSTRACT**

**CAB BOOKING SERVICES**

In the world where everyone prefers affordable and fast means of transportation, online platforms which help in booking vehicles have gained momentum off late. Google maps, ola and uber cabs provide a lot of services today that help in comfortable commute.

Our project deals with the prototype of a cab booking service. Firstly, the user is prompted to enter the source and destination. Then, a list of all the vehicles available is displayed to the user. The user can choose the vehicle based on the drivers rating, comfort zone and cost. After the booking is confirmed, a message is displayed showing the OTP and the price of the fare. The time at which the cab will reach the user’s current location will also be displayed. For this, we plan to use the shortest path algorithm. A city is a road network. It is assumed that the city has certain landmarks. Each of these landmarks is considered to be a node. This becomes a graph framework. The path that connects two nodes is assigned a value based on the traffic of the city at that particular time. So, based on that the shortest path is chosen by the driver and the amount of time required by the driver to reach his customer is displayed. When the destination is reached, the payment is done and the service is completed.

So, on the whole, we plan to use graphs that represent a road network, greedy approach to satisfy the user based on his needs, print the OTP and shortest path algorithm to display the time. Also, all pair shortest paths algorithm is used which is a part of dynamic programming.

**DESCRIPTION**

Our project deals with online cab booking services. We have considered 5 main places of the city in our road network. These places are Hi-tech city,Mehdipatnam , Kukatpally,Lakdikapul and Dilshuknagar. The road network is stored in the form of a graph.

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph.



Fig-1

The user is first displayed with the names of the places which are part of the road network. He is prompted to enter the pick up and drop location. After making the appropriate selection the user is asked to enter the number of passengers who want to avail the services. If the number of passengers is more than 1 then the user is provided the option for selecting either cab/auto. If not, the user has three choices. They are->cab, auto and bike. Then, the shortest path is calculated by using all pairs shortest path algorithm. Also, we provide cars for rent and based on the number of hours the cost is printed.

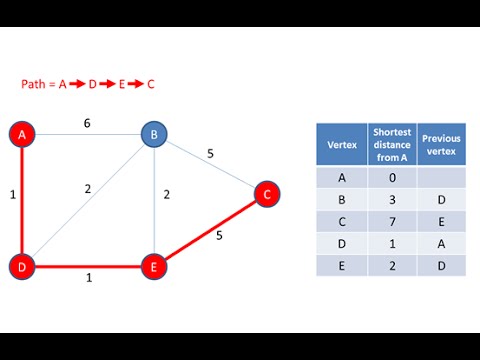


Fig-2

This is an algorithm with time complexity O(n^3) and it uses the concept of dynamic programming. The total fare is displayed based on the cost of each edge. The dijsktras algorithm is used to calculate the minimum time required for the vehicle to arrive so that the user can stay informed. Dijkstras algorithm makes use of greedy approach and the time complexity of the algorithm is O(n^2). This is also called as single source shortest path algorithm. The total fare for auto is half the rate of the total fare of the cab. The total fare of a bike is 1/3rd of the fare of the cab. In the end, the OTP which is a 4-digit number is generated and the booking is confirmed.

Data Structures and other concepts used are->

1. Graphs

2. Dynamic programming

3. Greedy approach

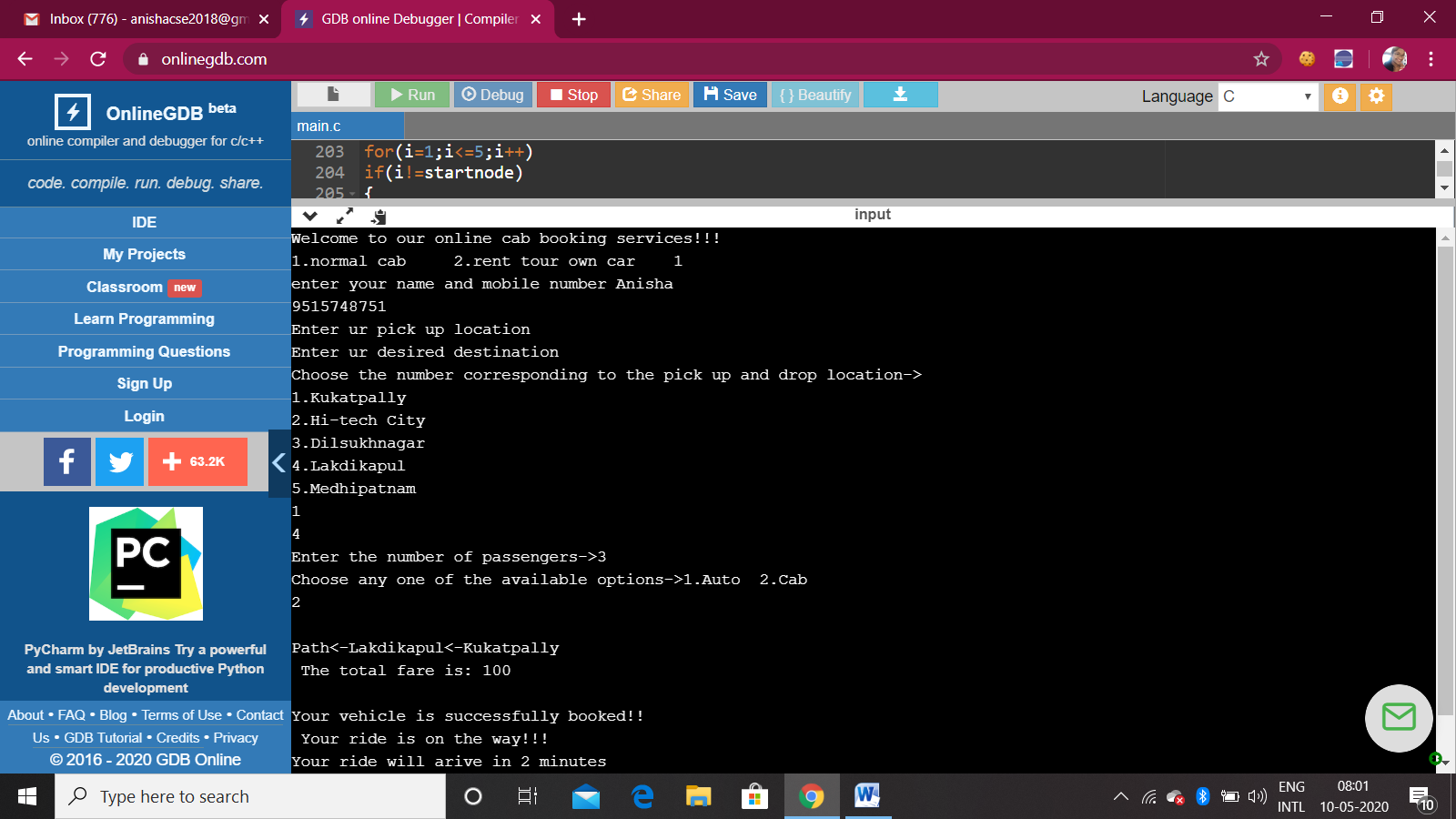
4. Random function in C language

5.Libraries used are stdio.h,time.h,stdlib.h,string.h

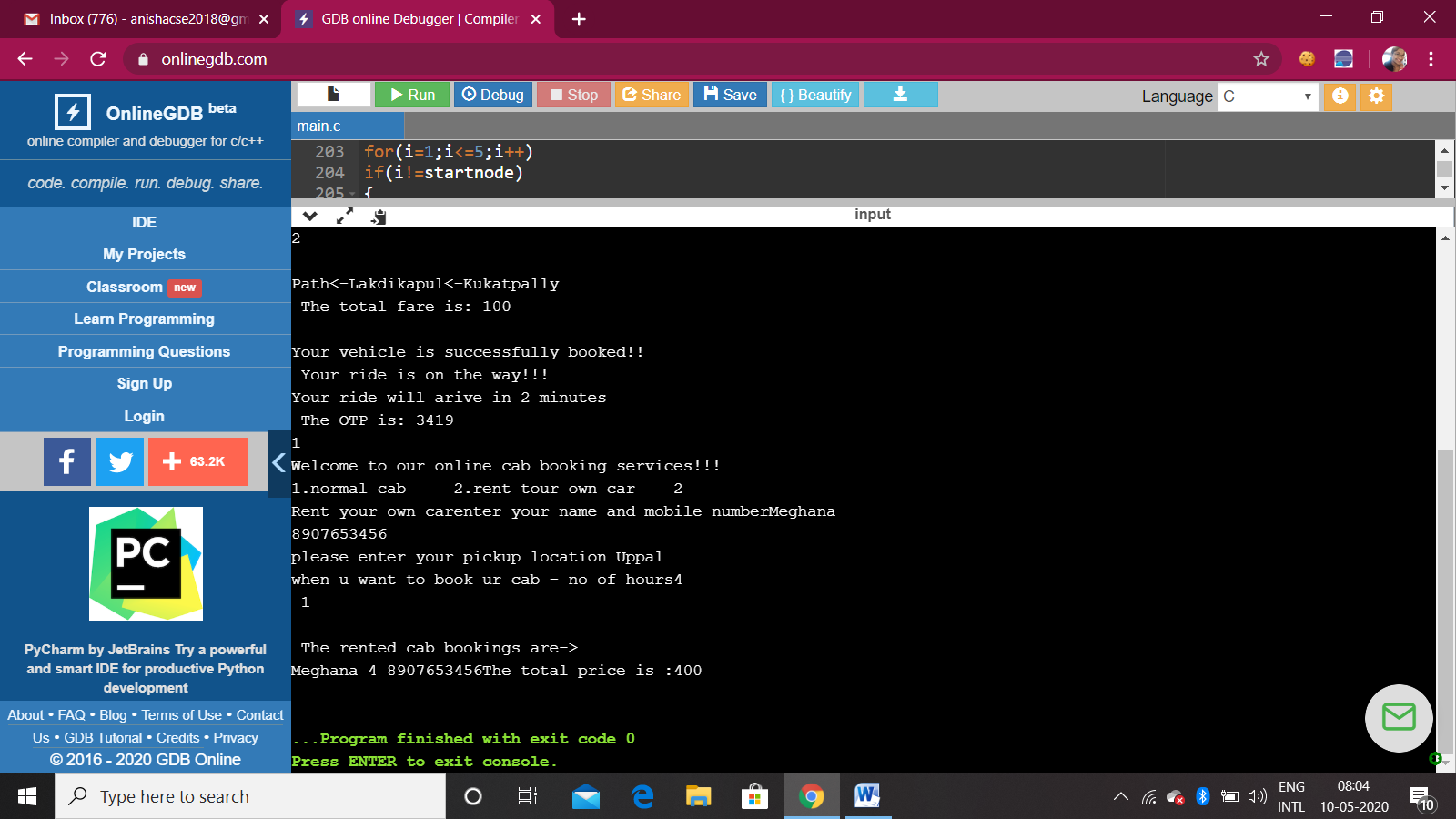
**CODE->**

**#include<stdio.h>  
#include<stdlib.h>  
#include<time.h>  
#include<string.h>  
int G[10][10],cost[10][10],pay[10][10],sum=0;  
int INF=999;  
struct next  
{  
    char name[100];  
    int hour;  
    long int number;  
}q[10];  
int min(int a,int b)  
{  
    return (a<b)?a:b;  
}  
void LocationsInCity()  
{  
    printf("1.Kukatpally\n");  
    printf("2.Hi-tech City\n");  
    printf("3.Dilsukhnagar\n");  
    printf("4.Lakdikapul\n");  
    printf("5.Medhipatnam\n");  
}  
void dijkstra(int cost[10][10],int startnode,int dst)  
{  
int v=dst;  
int distance[10],pred[10];  
int visited[10],count,mindistance,nextnode,i,j,sum=0;  
for(i=1;i<=5;i++)  
{  
for(j=1;j<=5;j++)  
{  
pay[i][j]=cost[i][j];  
}  
}  
for(i=1;i<=5;i++)  
{  
distance[i]=pay[startnode][i];  
pred[i]=startnode;  
visited[i]=0;  
}  
distance[startnode]=0;  
visited[startnode]=1;  
count=1;  
while(count<4)  
{  
mindistance=INF;  
for(i=1;i<=5;i++)  
if(distance[i]<mindistance&&!visited[i])  
{  
mindistance=distance[i];  
nextnode=i;  
}  
visited[nextnode]=1;  
for(i=1;i<=5;i++)  
if(!visited[i])  
if(mindistance+pay[nextnode][i]<distance[i])  
{  
distance[i]=mindistance+pay[nextnode][i];  
pred[i]=nextnode;  
}  
count++;  
}  
for(i=1;i<=5;i++)  
if(i!=startnode)  
{  
if(v==i){  
printf("\nPath");  
switch(i)  
{  
  case 1:{  
      printf("<-Kukatpally");  
      break;}  
  case 2:{  
      printf("<-Hi-tech City");  
      break;}  
  case 3:{  
      printf("<-Dilsukhnagar");  
      break;}  
  case 4:{  
      printf("<-Lakdikapul");  
      break;}  
  case 5:{  
       printf("<-Medhipatnam");  
      break;}  
}  
j=i;  
do  
{  
j=pred[j];  
switch(j)  
{  
  case 1:{  
      printf("<-Kukatpally");  
      break;}  
  case 2:{  
      printf("<-Hi-tech City");  
      break;}  
  case 3:{  
      printf("<-Dilsukhnagar");  
      break;}  
  case 4:{  
      printf("<-Lakdikapul");  
      break;}  
  case 5:{  
       printf("<-Medhipatnam");  
      break;}  
}  
//printf("\n");  
}while(j!=startnode);  
}}  
  
}  
int Map(int src,int dst)  
{  
    cost[1][1]=0;  
    cost[1][2]=100;  
    cost[1][3]=150;  
    cost[1][4]=100;  
    cost[1][5]=INF;  
    cost[2][1]=200;  
    cost[2][2]=0;  
    cost[2][3]=150;  
    cost[2][4]=INF;  
    cost[2][5]=INF;  
    cost[3][1]=INF;  
    cost[3][2]=INF;  
    cost[3][3]=0;  
    cost[3][4]=200;  
    cost[3][5]=150;  
    cost[4][1]=INF;  
    cost[4][2]=INF;  
    cost[4][3]=INF;  
    cost[4][4]=0;  
    cost[4][5]=100;  
    cost[5][1]=200;  
    cost[5][2]=INF;  
    cost[5][3]=INF;  
    cost[5][4]=INF;  
    cost[5][5]=0;  
    int i,j,k,a[10][10];  
    for(i=1;i<=5;i++)  
    {  
        for(j=1;j<=5;j++)  
        {  
            a[i][j]=cost[i][j];  
        }  
    }  
    for(k=1;k<=5;k++)  
    {  
        for(i=1;i<=5;i++)  
        {  
            for(j=1;j<=5;j++)  
            {  
                a[i][j]=min(a[i][j],a[i][k]+a[k][j]);  
            }  
        }  
    }  
    dijkstra(cost,src,dst);  
  return a[src][dst];  
}  
int dijkstra1(int G[10][10],int startnode,int dst)  
{  
int v=dst;int totaltime=0;  
int distance[10],pred[10];  
int visited[10],count,mindistance,nextnode,i,j,sum=0;  
for(i=1;i<=5;i++)  
{  
for(j=1;j<=5;j++)  
{  
pay[i][j]=G[i][j];  
}  
}  
for(i=1;i<=5;i++)  
{  
distance[i]=pay[startnode][i];  
pred[i]=startnode;  
visited[i]=0;  
}  
distance[startnode]=0;  
visited[startnode]=1;  
count=1;  
while(count<4)  
{  
mindistance=INF;  
for(i=1;i<=5;i++)  
if(distance[i]<mindistance&&!visited[i])  
{  
mindistance=distance[i];  
nextnode=i;  
}  
visited[nextnode]=1;  
for(i=1;i<=5;i++)  
if(!visited[i])  
if(mindistance+pay[nextnode][i]<distance[i])  
{  
distance[i]=mindistance+pay[nextnode][i];  
pred[i]=nextnode;  
}  
count++;  
}  
for(i=1;i<=5;i++)  
if(i!=startnode)  
{  
if(v==i){totaltime+=distance[i];  
j=i;  
do  
{  
j=pred[j];  
}while(j!=startnode);  
}}  
return totaltime;  
}  
int Time(int k,int src)  
{  
    G[1][1]=0;  
    G[1][2]=1;  
    G[1][3]=1;  
    G[1][4]=4;  
    G[1][5]=INF;  
    G[2][1]=2;  
    G[2][2]=0;  
    G[2][3]=3;  
    G[2][4]=INF;  
    G[2][5]=INF;  
    G[3][1]=INF;  
    G[3][2]=INF;  
    G[3][3]=0;  
    G[3][4]=2;  
    G[3][5]=4;  
    G[4][1]=INF;  
    G[4][2]=INF;  
    G[4][3]=INF;  
    G[4][4]=0;  
    G[4][5]=3;  
    G[5][1]=2;  
    G[5][2]=INF;  
    G[5][3]=INF;  
    G[5][4]=INF;  
    G[5][5]=0;  
    dijkstra1(G,k,src);  
}  
int printRandoms(int lower, int upper,int count)  
{  
    int i;  
    for (i = 0; i < count; i++) {  
        int num = (rand() %  
           (upper - lower + 1)) + lower;  
        return num;  
    }  
}  
int main() {  
 int source,destination,n,passengers,choice,ri,time1,k,p,w,noofreq,xx;  
 int lower = 1, upper = 5, count = 1,temp,kb,x=0,f,i,j,r,flag=0;  
 int d[10],J[10];  
 char c[100],h[100];  
 char copy[100];  
 long int yy,m;  
 int u=1;  
 while(u!=-1)  
 {printf("Welcome to our online cab booking services!!!\n");  
 printf("1.normal cab\t 2.rent tour own car\t");  
 scanf("%d",&w);  
 switch(w)  
 {  
 case 1:{  
    printf("enter your name and mobile number");  
 scanf("%s %ld",c,&m);  
 printf("Enter ur pick up location\n");  
 printf("Enter ur desired destination\n");  
 printf("Choose the number corresponding to the pick up and drop location->\n");  
 LocationsInCity();  
 scanf("%d%d",&source,&destination);  
 printf("Enter the number of passengers->");  
 scanf("%d",&passengers);  
 if(passengers==1)  
 {  
     printf("Choose any one of the available options->");  
     printf("Press 1.Auto\t 2.Bike\t 3.Cab\n");  
     scanf("%d",&choice);  
     r=Map(source,destination);  
     switch(choice)  
     {  
         
         case 1:{  
                 printf("\n The total fare is : %d\n",r/2);  
                 
                 break;}  
         case 2:{  
             printf("\n The total fare is: %d\n",r/3);  
             break;}  
         case 3:{  
             printf("\n The total fare is: %d\n",r);  
             break;}  
     }  
 }  
 if(passengers>=2 && passengers<=4)  
 {  
     printf("Choose any one of the available options->");  
     printf("1.Auto\t 2.Cab\n");  
     scanf("%d",&choice);  
     r=Map(source,destination);  
     switch(choice)  
     {  
           
         case 1:{printf("\n The total fare is: %d\n",r/2);break;}  
         case 2:{printf("\n The total fare is: %d\n",r); break;}  
     }  
 }  
 else{  
     printf("Too many passengers cannot be accomadated!!");  
 }  
 printf("\nYour vehicle is successfully booked!!");  
 printf("\n Your ride is on the way!!!");  
   srand(time(0));  
  k=printRandoms(lower, upper, count);  
  time1=Time(k,source);  
  if(time1==0)  
  {  
      printf("\n Your ride is here!!!");  
  }  
  else{  
 printf("\nYour ride will arive in %d minutes",time1);  
  }  
  srand(time(0));  
  p=printRandoms(1000,9999,1);  
  printf("\n The OTP is: %d\n",p);  
  break;  
 }  
    case 2:  
    {     printf("Rent your own car");  
           printf("enter your name and mobile number");  
           scanf("%s%ld",c,&m);  
           x=x+1;  
           flag=1;  
           printf("please enter your pickup location");  
           scanf("%s",h);  
           printf("when u want to book ur cab - no of hours");  
           scanf("%d",&f);  
           q[x-1].hour=f;  
           strcpy(q[x-1].name,c);  
           q[x-1].number=m;  
          for(i=0;i<x;i++)  
          {  
           for(j=0;j<x-i-1;j++)  
           {  
             if(q[j].hour<q[j+1].hour)  
             {  
                 temp=q[j].hour;  
                 q[j].hour=q[j+1].hour;  
                 q[j+1].hour=temp;  
                  strcpy(copy,q[j].name);  
                 strcpy(q[j].name,q[j+1].name);  
                 strcpy(q[j+1].name,copy);  
                 yy=q[j].number;  
                 q[j].number=q[j+1].number;  
                 q[j+1].number=yy;  
             }  
           }  
         }  
      break;      
    }  
 }  
 scanf("%d",&u);  
}  
if(flag==1)  
{  
         for(i=0;i<x;i++)  
         {  
             printf("\n The rented cab bookings are->\n");  
             printf("%s %d %ld",q[i].name,q[i].hour,q[i].number);**

**printf("The total price is :%d\n",q[i].hour\*100);  
         }  
}  
 return 0;  
}**

**OUTPUT->**

The above output shows cab booking service. Firstly, the user is prompted to enter his details along with the pick- up and drop location. Based on the places selected, the path is chosen and is displayed. Based on the distance, the total fare is displayed. The OTP is generated and the booking is confirmed.

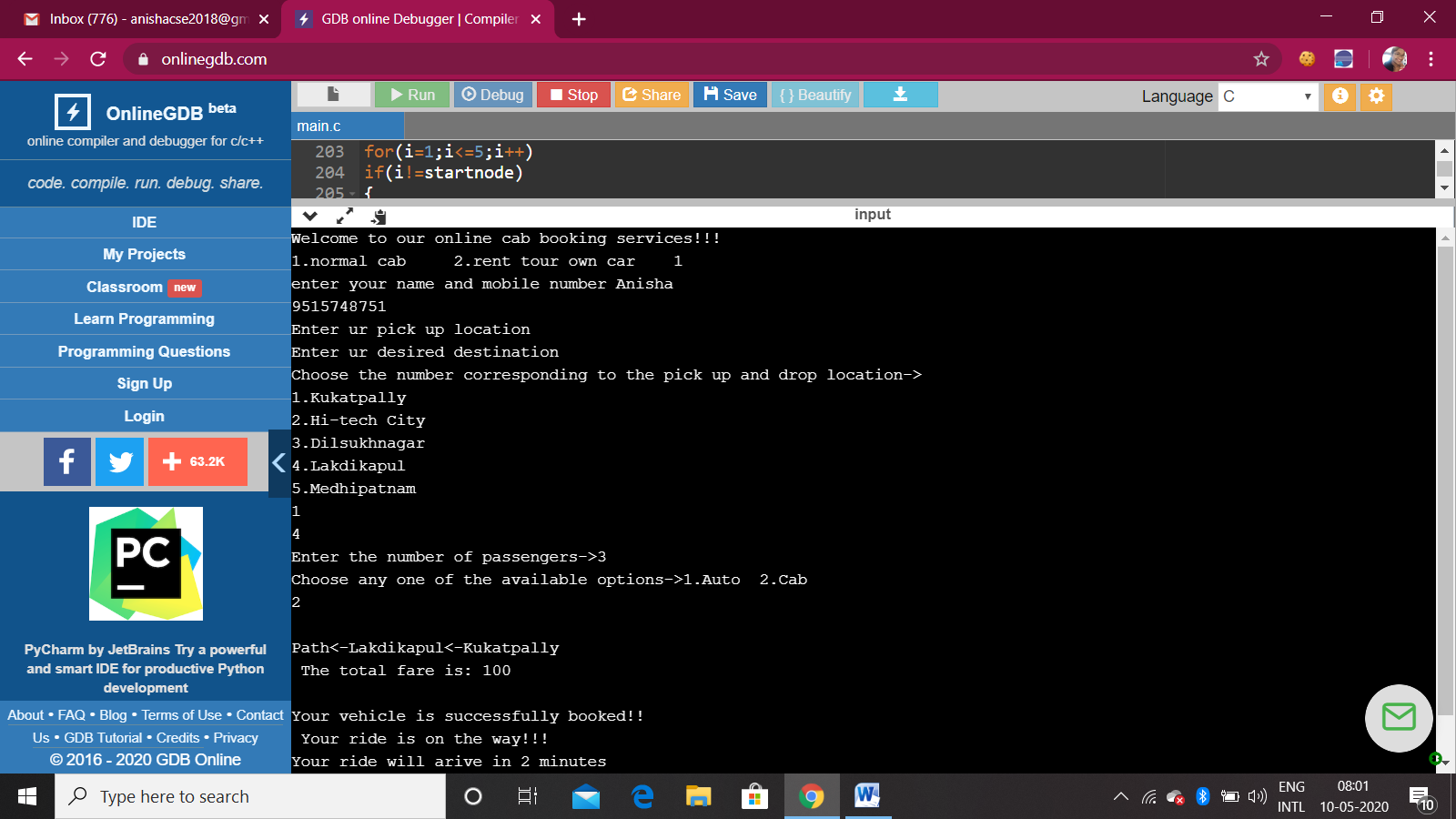


The above output is for renting a cab. The user has to fill his/her personal details and select the appropriate location from where he wants to rent the cab. The total cost is printed based on the number of hours he wants to rent the cab.

**CONSTRAINTS**

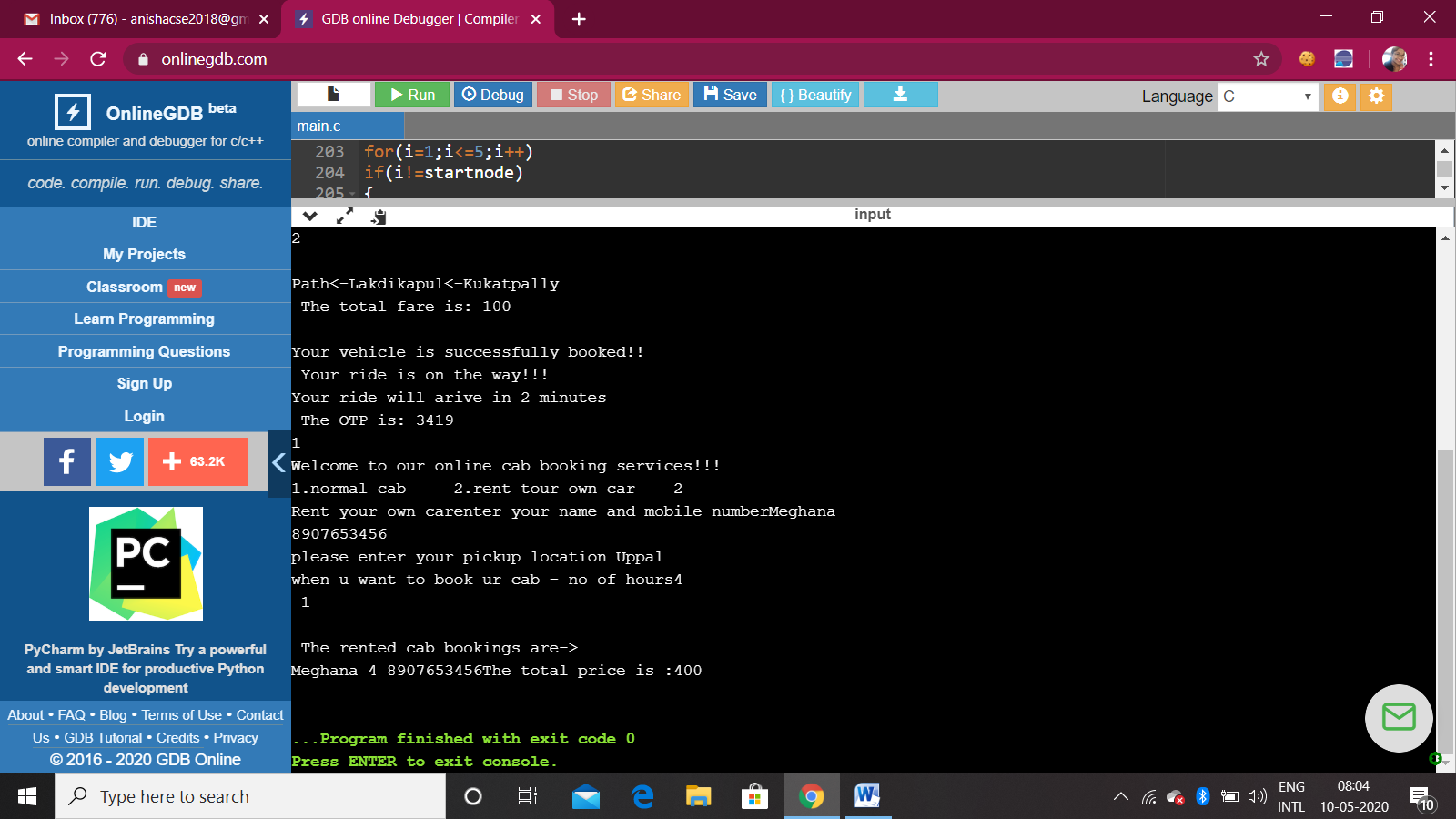
1. The number of places in the city road network is 5.
2. At which ever time of the day, the vehicle is booked, it will arrive in at most 5 minutes.

**TEST CASES**

1. **Normal cab booking services** :

The above output shows cab booking service. Firstly, the user is prompted to enter his details along with the pick- up and drop location. Based on the places selected, the path is chosen and is displayed. Based on the distance, the total fare is displayed. The OTP is generated and the booking is confirmed.

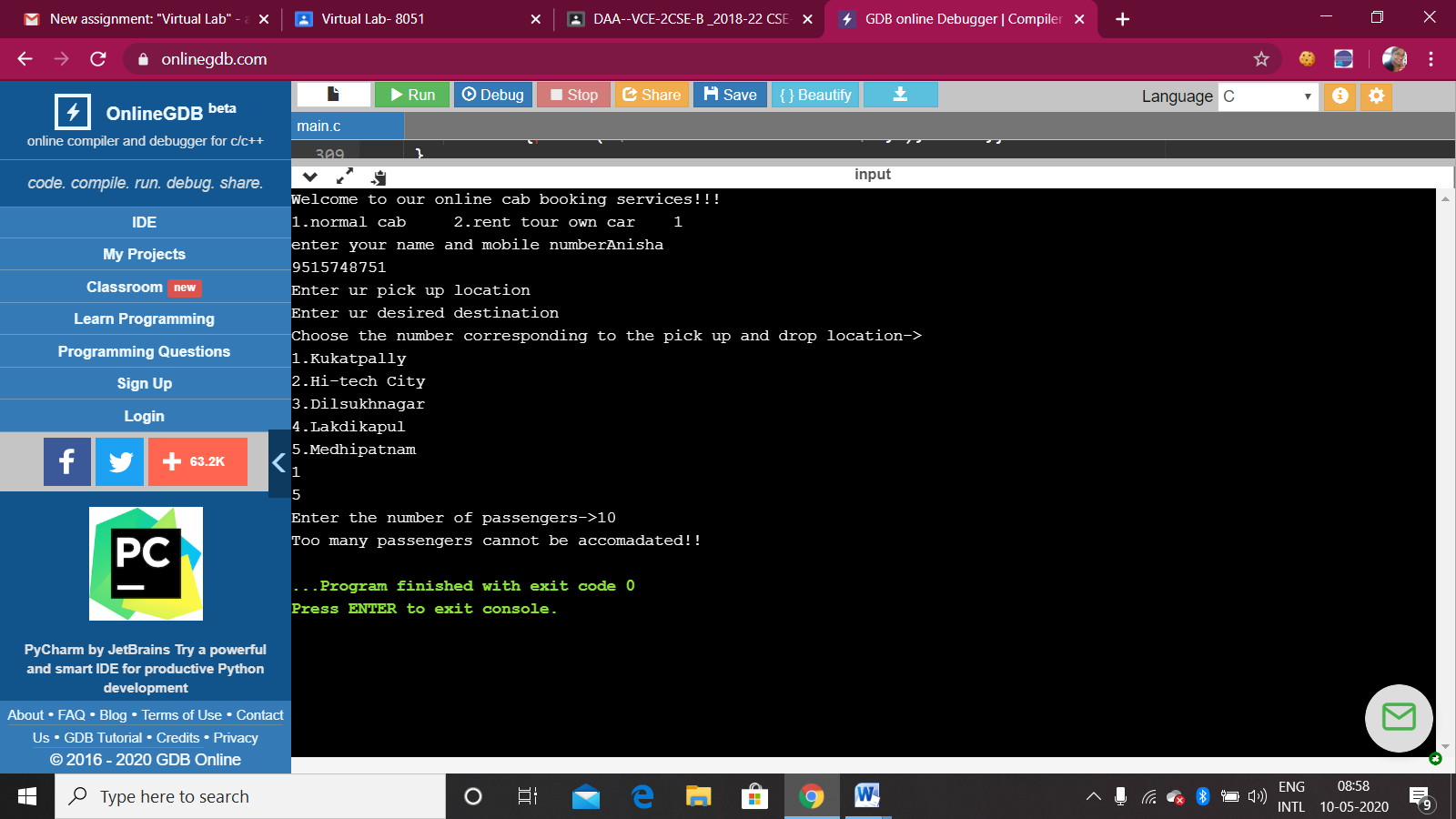
**2. Rental cab:**



The above output is for renting a cab. The user has to fill his/her personal details and select the appropriate location from where he wants to rent the cab. The total cost is printed based on the number of hours he wants to rent the cab.

**3. Constraints on the number of people:**

The total number of passengers are less than 4, only then , they can travel by auto or cab. If it exceeds this count, those many people cannot be accommodated. This message is displayed and the program gets exited. The below output screenshot describes the same**.**



**CONCLUSION**

The project deals with online cab booking services. In the 21st century, where everyone desires speedy results, these cab booking services help in faster transportation. The design strategies used for its implementation are dynamic programming and greedy strategy. The algorithms used are all pair shortest paths and dijkstra’s algorithm.

**REFERENCES->**

|  |
| --- |
| https://mail.google.com/mail/u/0/images/cleardot.gif  https://mail.google.com/mail/u/0/images/cleardot.gif |
|

* <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>
* <https://www.tutorialspoint.com/c_standard_library/c_function_rand.html>
* TEXTBOOK REFERRED- DESIGN AND ANALYSIS OF ALGORITHMS BY ELLIS HOROWITZ AND SATRAJ SAHANI