DELHI METRO ROUTE FINDER

CSE2003 – Data Structures and algorithms

Name of the student: YUKTI KHOLIWAL

Name of the faculty: Prof. Sanjiban Sekhar Roy

Slot: A2

Abstract

The Delhi Metro is a mass rapid transit system serving Delhi and its satellite cities. It is by far the largest and busiest metro rail system in India. The network consists of 10 color coded lines serving 253 stations in Delhi with a total length of 348 kilometers and has a full reach of 285 stations. Since it is widely spread, a person constantly needs to change metro lines to reach destination station. Dijkstra's algorithm solves the problem of finding the shortest path from a point in a graph (the source station) to a destination station. The project focuses on giving the sequential path from source to destination also mentioning where to change the metro line and towards which station. The algorithm also counts the number of stations a person will have to travel.

Motivation

As someone who lives in Delhi, Delhi metro is the easiest, fastest and cheapest way to go to any place in the city. It's a tedious task to look at the metro map , for the change of routes, metro line, different stations, every time you want to travel to a different location. So to make this task easier for the passengers, I decided to make this project.

Objective

To make a program using the concepts of data structures and algorithms which solves this problem. The program will ask the user to enter the current metro station (source location) and the destination metro station. The program will find the shortest route to the destination by providing all the metro stations and where to change the metro line and towards which direction. It will also show the number of stations.

Algorithm used: Dijkstra's algorithm

What is Dijkstra's algorithm?

- Dijkstra algorithm is a greedy algorithm.
- It finds the shortest paths between nodes in a graph, which may represent, for example, road networks
- For a given source node in the graph, the algorithm finds the shortest path between the source node and every other node.
- This algorithm also used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.
- Dijkstra's Algorithm basically starts at the node that you choose (the source node) and it analyzes the graph to find the shortest path between that node and all the other nodes in the graph.

- The algorithm keeps track of the currently known shortest distance from each node to the source node and it updates these values if it finds a shorter path.
- Once the algorithm has found the shortest path between the source node and another node, that node is marked as "visited" and added to the path.
- The process continues until all the nodes in the graph have been added to the path. This way, we have a path that connects the source node to all other nodes following the shortest path possible to reach each node.
- Dijkstra's algorithm uses a priority queue data structure.

Pseudo code of the algorithm

```
1   function Dijkstra(Graph, source):
3       create vertex set Q
5       for each vertex v in Graph:
6           dist[v] ← INFINITY
7           prev[v] ← UNDEFINED
8           add v to Q
9       dist[source] ← 0
11      while Q is not empty:
12          u ← vertex in Q with min dist[u]
14          remove u from Q
16          for each neighbor v of u:          // only v that are still in Q
17              alt ← dist[u] + length(u, v)
18              if alt < dist[v]:
19                  dist[v] ← alt
20                  prev[v] ← u
22      return dist[], prev[]
```

The execution of Dijkstra's algorithm can be seen in the below image

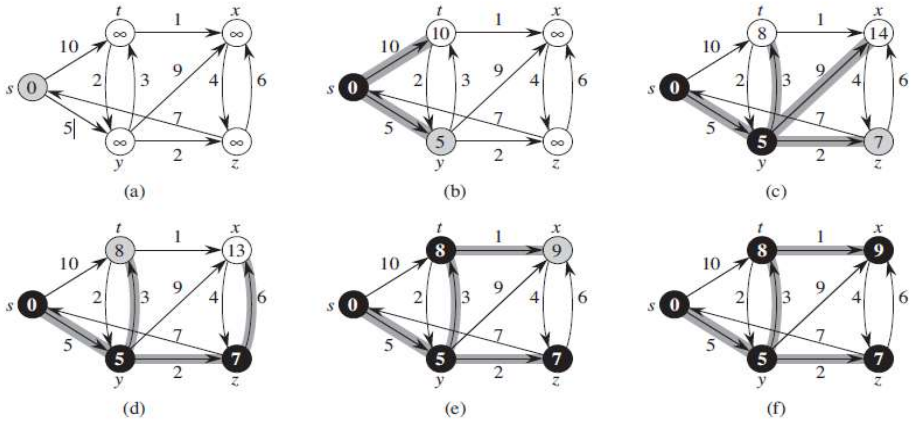
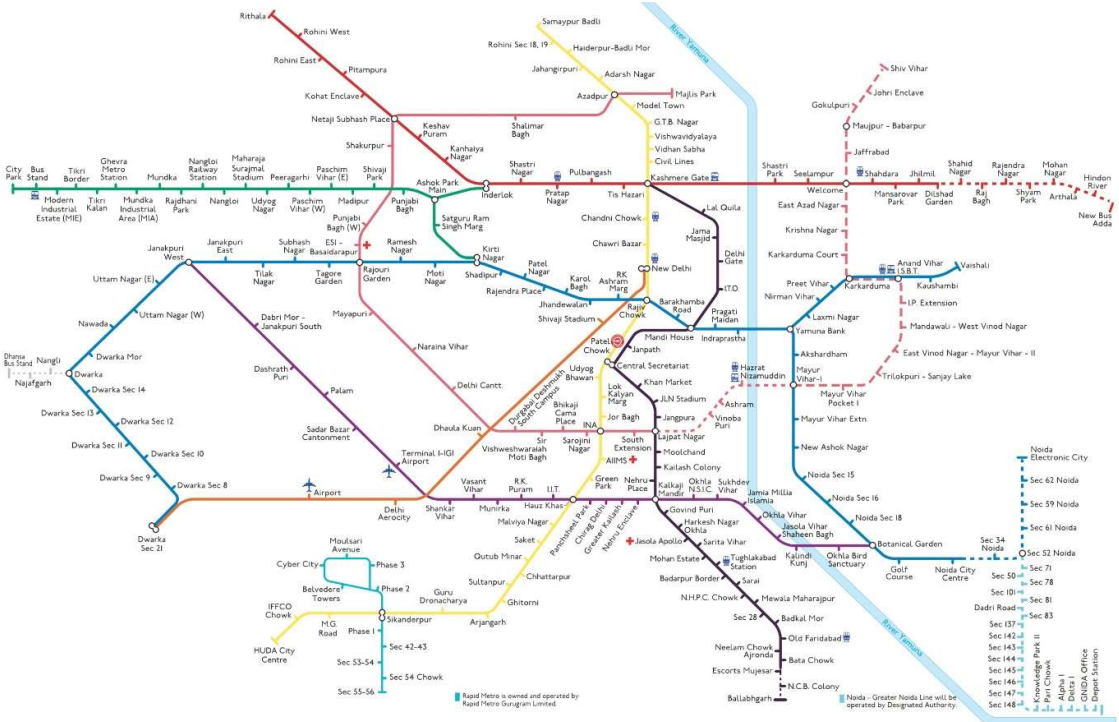


Figure 24.6 The execution of Dijkstra's algorithm. The source s is the leftmost vertex. The shortest-path estimates appear within the vertices, and shaded edges indicate predecessor values. Black vertices are in the set S, and white vertices are in the min-priority queue $Q = V - S$. **(a)** The situation just before the first iteration of the **while** loop of lines 4–8. The shaded vertex has the minimum d value and is chosen as vertex u in line 5. **(b)–(f)** The situation after each successive iteration of the **while** loop. The shaded vertex in each part is chosen as vertex u in line 5 of the next iteration. The d values and predecessors shown in part (f) are the final values.

THE METRO MAP OF DELHI



STEPS INVOLVED IN THE IMPLEMENTATION OF THE PROGRAM

- Finding out Source number and destination number of the source and destination stations
- Finding out Source color and destination color of the source and destination stations
- Solving the junction conflicts
- Building Adjacency/Connectivity Matrix of all the metro stations lying in blue, yellow and violet metro line. All the stations are nodes in the graph.
- Applying dijkstra algorithm to find the shortest path
- Using findPath() function whose input is source point and destination point and output is list of stations from source to destination and also the number of stations travelled.

Source number and destination number decider: Station() function specifies a unique number to all the stations. Entered station name is compared with each station in this function and give its number.

```
/*Entered station name is compared with each stations in this function*/
int station(char Name[30])
{
int i;
//yellow line
i=strcasecmp(Name,"Huda City Center");
if(i==0)
return 0;
i=strcasecmp(Name,"Iffco Chowk");
if(i==0)
return 1;
i=strcasecmp(Name,"MG Road");
if(i==0)
return 2;
i=strcasecmp(Name,"Sikanderpur");
if(i==0)
return 3;
i=strcasecmp(Name,"Guru Dronacharya");
if(i==0)
return 4;
i=strcasecmp(Name,"Arjangarh");
```

Source color and destination color: colorDecider() function uses the source number or the destination number obtained in the previous step and gives the metro line color of the source and destination station.

```
int colorDecider(int Number)
{
int Color;
if(Number>=0 && Number <=37)
Color = Yellow;
else if(Number>=38 && Number <=89)
Color = Blue;
else if(Number>=130 && Number <=157)
Color = Violet;
return Color;
}
```

Solving the junction conflicts: At the junctions where there are multiple number and colors. To solve that part:

```
/*There are some conflicts in junctions, This part solves that*/
if(sourceNumber == 21 && destColor !=Yellow)//Junction conflict Yellow/blue line
{
sourceNumber = 66;
sourceColor = Blue;
}
if(destNumber == 21 && sourceColor !=Yellow)//Junction conflict Yellow/blue line
{
destNumber = 66;
destColor = Blue;
}
if(sourceNumber == 68 && destColor ==Violet)//Junction conflict Violet/blue line
{
sourceNumber = 131;
sourceColor = Violet;
}
if(destNumber == 68 && sourceColor ==Violet)//Junction conflict Violet/blue line
{
destNumber = 131;
sourceColor = Violet;
```

Creating the graph using adjacency matrix: Next step is to use createGraph() function. This function will create the graph of all the stations as the nodes and connecting them via edges.

```
void createGraph()//This function connects stations according to Delhi Metro routes
{
adj[0][1]=4;
adj[1][0]=4;
adj[1][2]=4;
adj[2][1]=4;
adj[2][3]=4;
adj[3][2]=4;
adj[3][4]=4;
adj[4][3]=4;
adj[4][5]=4;
adj[5][4]=4;
adj[5][6]=4;
adj[6][5]=4;
adj[6][7]=4;
adj[7][6]=4;
adj[7][8]=4;
adj[8][7]=4;
adj[8][9]=4;
adj[9][8]=4;
```

Using Dijkstra's Algorithm: After creating the graph, we will use dijikstra's algorithm to find the shortest route between the source and destination station.

```
void dijkstra(int sourceNumber)
{
int i,n=200,current;

    for(i=0; i<n; i++)
    {
        predecessor[i] =  NIL;
        pathLength[i] = infinity;
        status[i] = TEMP;
    }
    pathLength[sourceNumber] = 0;

    while(1)
    {
        current = min_temp( );
        if( current == NIL )
            return;
        status[current] = PERM;
        for(i=0; i<n; i++)
        {
            if ( adj[current][i] !=0 && status[i] == TEMP )
                if( pathLength[current] + adj[current][i] < pathLength[i] )
                {
                    predecessor[i] = current;
                    pathLength[i] = pathLength[current] + adj[current][i];
                }
        }
    }
}/*End of Dijkstra( )*/
```

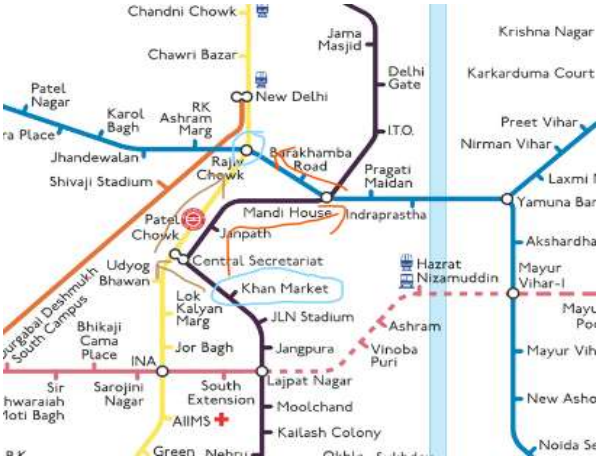
Finding the path from source to destination: findPath() function will take input as sorce number, destination number and source color and will give us the path and number of stations we will have to travel from source to destination.

```
void findPath(int sourceNumber, int destNumber,int sourceColor )
{
    int i,u,j,a,joe,t=-11;
    int path[MAX];
    int shortdist = 0;
    int count = 0;
    int path2[MAX];
    int firststation;
    int Nextstation;
    int vb;
    /*Store the full path in the array path*/
    while( destNumber != sourceNumber )
    {
        count++;
        path[count] = destNumber;
        u = predecessor[destNumber];
                shortdist = count;
        destNumber = u;
    }
    count++;
    shortdist++;
    path[count]=sourceNumber;
        for(i=count,j=1;i>=1,j<=count;i--,j++)
        path2[j]=path[i];
        firststation = path2[1];
        Nextstation = path2[2];
        if((firststation - Nextstation)==1)
         {

        else if((firststation - Nextstation)==-1)
         {
          printf("\nBoard on a metro heading towards ");
          rightSame(sourceColor);
         }
    for(i=1; i<=count; i++)
               {
                a=t;
                t = path2[i];

                joe=joLeRi(a,t);
                if(joe==1)
                {
                shortdist=shortdist-1;
                vb=path2[i+1];
                headingWhere(vb);
                joe=0;
                }
                if(joe==2)
                {
                shortdist=shortdist-1;
                vb=path2[i+1];
                headingWhere2(vb);
                joe=0;
                }
                numToName(t);
               }

        printf("Deboard here(This is your Destination)\n");
    printf("\nNumber of stations are : %d\n",shortdist);
}/*End of findPath()*/
```

Program Output

Sample Output-1:

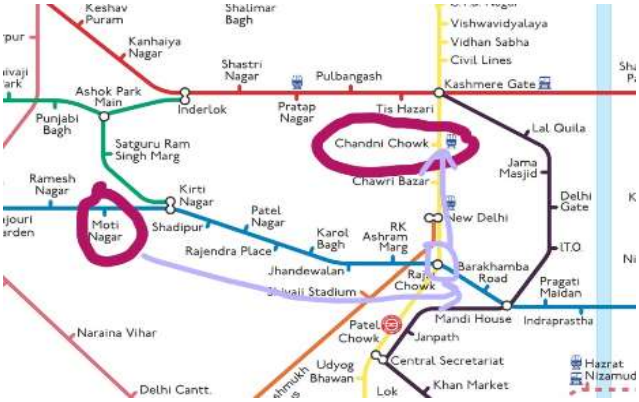
```
"C:\Users\The Philocalist\Documents\da\review2.exe"

Enter the Source Station Name
Khan Market

Enter the Destination Station Name
Rajiv chowk

Board on a metro heading towards ITO
Khan Market->
Central Secratariat->
Change here for Yellow Line

Board on a metro heading towards Samaypur Badli
Central Secrateriat->
Patel Chowk->
Rajiv Chowk->
Change here for Blue Line

Rajiv Chowk->Deboard here(This is your Destination)

Number of stations are : 4

Process returned 0 (0x0)   execution time : 21.969 s
Press any key to continue.
```



- Travelling from Khan Market to Rajiv Chowk. As you can see in the above image, there are 2 ways to travel between them.
- First way: Khan Market ->Mandi House, then here change for blue line towards Dwarka and then Mandi House -> Rajiv Chowk. It will take 6 stations to reach the destination.
- Second way: Kan Market ->Central Secretariat, over here change for yellow line towards Samypur Badli and then Central Secretariat -> Rajiv Chowk. Here the number of stations are 4.
- Since the latter route has less number of stations, therefore our program output will be the second way.

Sample Output-2:

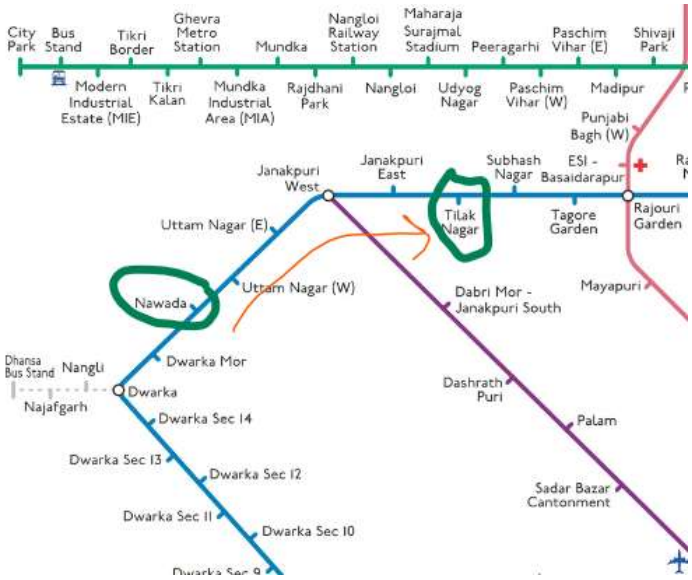
```
"C:\Users\The Philocalist\Documents\da\review2.exe"

Enter the Source Station Name
Moti Nagar

Enter the Destination Station Name
Chandni chowk

Board on a metro heading towards Noida City Center/Vaishali

Moti Nagar->
Kirti Nagar->
Shadipur->
Patel Nagar->
Rajendra Place->
Karol Bagh->
Jhandewalan->
Ramakrishna Ashram Marg->
Rajiv Chowk->
Change here for Yellow Line

Board on a metro heading towards Samaypur Badli

Rajiv Chowk->
New Delhi->
Chawri Bazaar->
Chandni Chowk->Deboard here(This is your Destination)

Number of stations are : 12

Process returned 0 (0x0)   execution time : 15.638 s
Press any key to continue.
```



- In the above we entered the source station name as Moti Nagar and destination station name as Chandni Chowk. As you can see in the above image that these two metro stations are on different metro lines. Our program is showing us that we have to board a metro heading towards Vaishali/Noida city center. The output is showing all the stations that are coming on the way.
- Now on Rajiv Chowk metro station, the program output shows to change the metro line and go for yellow line metro towards Samypur Badli. Now after a couple of stations we will reach our destination station.
- The output of the program also shows the number of stations we will be crossing to reach the destination station.

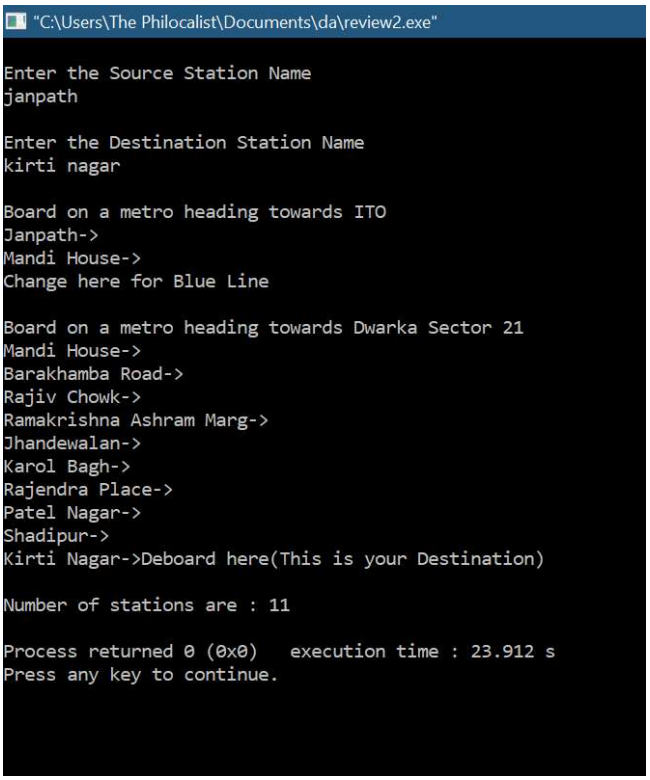
Sample Output-3:

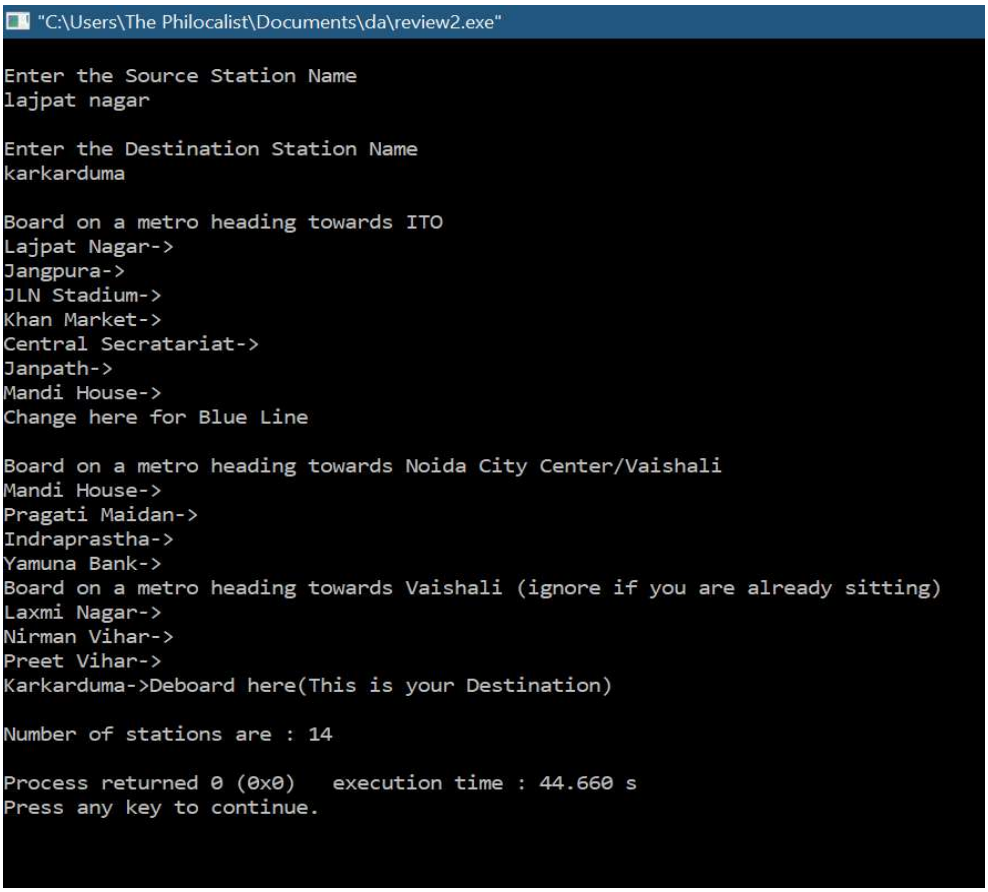
```
"C:\Users\The Philocalist\Documents\da\review2.exe"

Enter the Source Station Name
Nawada

Enter the Destination Station Name
Tilak Nagar

Board on a metro heading towards Noida City Center/Vaishali

Nawada->
Uttam Nagar West->
Uttam Nagar East->
Janakpuri West
Janakpuri East
Tilak Nagar->Deboard here(This is your Destination)

Number of stations are : 6

Process returned 0 (0x0)   execution time : 12.010 s
Press any key to continue.
```



In the above example, the passenger travels between stations on same metro line i.e blue line. The output shows all the stations the passenger has to travel to reach the destination station. It also shows the number of stations the passenger will be travelling.

Some more sample outputs:





Conclusions: From the above examples it can be concluded that Dijkstra's algorithm can be used to find the shortest route between source and destination station. Our code displays when to change metro lines, which all stations will come in the journey and also the number of stations that the metro will be travelling.

References:

[1] Y. Cao, The Shortest path algorithm in data structures. Yibin University, vol. 6, 2007,pp.82 -84.

[2] Taha, H. Operations research an introduction, ninth edition. Pearson publisher, 2011.

[3] Nar, D. Graph theory with applications to engineering and computer science, Prentice Hall, 1997.

[4] Sneyers J., Schrijvers, T. and Demoen B. Dijkstra's Algorithm with Fibonacci Heaps: An Executable

Description in CHR. 20th Workshop on Logic Programming (WLP'06), Vienna, Austria, February 2006.

[5] Ravi, N., Sireesha, V. Using Modified Dijkstra's Algorithm for Critical Path Method in a Project Network.

CODE OF THE PROGRAM:

```
#include<stdio.h>
#include<strings.h>
#include<stdlib.h>
#define Yellow 1001
#define Blue 1002
#define Violet 1005
#define MAX 200
#define MAXMATCH 15
#define infinity 9999
#define NIL -1
#define TEMP 0
#define PERM 1
int checkJunc;
int adj[MAX][MAX];
int predecessor[MAX];
int pathLength[MAX];
int status[MAX];
int Match3[MAXMATCH];
int top=-1;
void push3(int data);
int pop3();
int station(char Name[30]); //Getting designated number of the source/destination names
void leftSame(int destColor);
void rightSame(int destColor);
void createGraph();
void dijkstra(int sourceNumber);
int min_temp();
void findPath(int sourceNumber,int destNumber,int sourceColor);
int joLeRi(int a,int t);
void numToName(int t);
void headingWhere(int vb);
void headingWhere2(int vb);
int colorDecider(int Number);
int main()
{
int sourceColor,destColor,sourceNumber,destNumber,item=-10,manualcheck;
char sourceName[30];
char destName[30];
/*source station part begins here*/
printf("\nEnter the Source Station Name\n");
gets(sourceName);
sourceNumber = station(sourceName);
//Source Line decider
sourceColor = colorDecider(sourceNumber);
/*Source part ends here*/
/*Destination part begins from here*/
printf("\nEnter the Destination Station Name\n");
scanf(" %30[^\n]%*c",destName);
destNumber = station(destName);
if(destNumber==sourceNumber)
{
printf("\nSource and Destination Stations are the same\n");
exit(1);
}
```

```
//Destination line decider
destColor = colorDecider(destNumber);
/*Destination part ends here*/
/*There are some conflicts in junctions, This part solves that*/
if(sourceNumber == 21 && destColor !=Yellow)//Junction conflict Yellow/blue line
{
sourceNumber = 66;
sourceColor = Blue;
}
if(destNumber == 21 && sourceColor !=Yellow)//Junction conflict Yellow/blue line
{
destNumber = 66;
destColor = Blue;
}
if(sourceNumber == 68 && destColor ==Violet)//Junction conflict Violet/blue line
{
sourceNumber = 131;
sourceColor = Violet;
}
if(destNumber == 68 && sourceColor ==Violet)//Junction conflict Violet/blue line
{
destNumber = 131;
sourceColor = Violet;
}
if(destNumber == 19 && sourceColor ==Blue)//Junction conflict Yellow/Violet line
{
destNumber = 133;
destColor = Violet;
}
if(sourceNumber == 19 && destColor ==Violet)//Junction conflict Yellow/Violet line
{
sourceNumber = 133;
sourceColor = Violet;
}
if(sourceNumber == 68 && destColor ==Yellow)//Junction conflict Violet/blue line
{
if(destNumber<=20)
{
sourceNumber = 131;
sourceColor = Violet;
}
}
if(sourceNumber == 19 && destColor ==Blue)//Junction conflict Violet/blue line
{
if(destNumber >=67 && destNumber<90)
{
sourceNumber = 133;
sourceColor = Violet;
}
}
/*Junction conflicts end here*/
createGraph();
dijkstra(sourceNumber);
findPath(sourceNumber,destNumber,sourceColor);
}
void dijkstra(int sourceNumber)
{
```

```
int i,n=200,current;
        for(i=0; i<n; i++)
        {
                predecessor[i] =  NIL;
                pathLength[i] = infinity;
                status[i] = TEMP;
        }
        pathLength[sourceNumber] = 0;
        while(1)
        {
                current = min_temp( );
                if( current == NIL )
                        return;
                status[current] = PERM;
                for(i=0; i<n; i++)
                {
                        if ( adj[current][i] !=0 && status[i] == TEMP )
                                if( pathLength[current] + adj[current][i] < pathLength[i] )
                                {
                                        predecessor[i] = current;
                                        pathLength[i] = pathLength[current] + adj[current][i];
                                }
                }
        }
}/*End of Dijkstra( )*/

int min_temp( )
{
        int i;
   int n=200;
        int min = infinity;
        int k = NIL;
        for(i=0;i<n;i++)
        {
                if(status[i] == TEMP && pathLength[i] < min)
                {
                        min = pathLength[i];
                        k = i;
                }
        }
        return k;
}/*End of min_temp( )*/

void findPath(int sourceNumber, int destNumber,int sourceColor )
{
        int i,u,j,a,joe,t=-11;
        int path[MAX];
        int shortdist = 0;
        int count = 0;
   int path2[MAX];
   int firststation;
   int Nextstation;
   int vb;
        /*Store the full path in the array path*/
        while( destNumber != sourceNumber )
        {
                count++;
```

```
                path[count] = destNumber;
                u = predecessor[destNumber];
        shortdist = count;
                destNumber = u;
        }
        count++;
  shortdist++;
        path[count]=sourceNumber;
    for(i=count,j=1;i>=1,j<=count;i--,j++)
    path2[j]=path[i];
    firststation = path2[1];
    Nextstation = path2[2];
    if((firststation - Nextstation)==1)
     {
      printf("\nBoard on a metro heading towards ");
      leftSame(sourceColor);
     }
    else if((firststation - Nextstation)==-1)
     {
      printf("\nBoard on a metro heading towards ");
      rightSame(sourceColor);
     }
        for(i=1; i<=count; i++)
        {
        a=t;
        t = path2[i];

        joe=joLeRi(a,t);
        if(joe==1)
        {
        shortdist=shortdist-1;
        vb=path2[i+1];
        headingWhere(vb);
        joe=0;
        }
        if(joe==2)
        {
        shortdist=shortdist-1;
        vb=path2[i+1];
        headingWhere2(vb);
        joe=0;
        }
        numToName(t);
        }

    printf("Deboard here(This is your Destination)\n");
        printf("\nNumber of stations are : %d\n",shortdist);
}/*End of findPath()*/

/*This function tells us where to change Metro*/
int joLeRi(int a,int t)
{
if(a==21 && t==66)
{
printf("\nChange here for Blue Line\n");
return 1;
}
```

```
else if(a==66 && t==21)
{
printf("\nChange here for Yellow Line\n");
return 1;
}
else if(a==71 && t==73)
{
printf("\nBoard on a metro heading towards Noida city Center (ignore if you are already sitting)");
return 0;
}
else if(a==71 && t==83)
{
printf("\nBoard on a metro heading towards Vaishali (ignore if you are already sitting)");
return 0;
}
else if(a==128 && t==59)
{
printf("\nChange here for Blue Line\n");
return 1;
}
else if(a==73 && t==71)
{
return 0;
}
else if(a==83 && t==71)
{
return 0;
}
else if(a==104 && t==25)
{
printf("\nChange here for Yellow Line\n");
return 1;
}
else if(a==68 && t==131)
{
printf("\nChange here for Violet Line\n");
return 2;
}
else if(a==131 && t==68)
{
printf("\nChange here for Blue Line\n");
return 2;
}
else if(a==19 && t==133)
{
printf("\nChange here for Violet Line\n");
return 1;
}
else if(a==133 && t==19)
{
printf("\nChange here for Yellow Line\n");
return 2;
}
else if(a==19 && t==134)
{
printf("\nChange here for Violet Line\n");
return 2;
```

```
}
else
return 0;
}
/*End of joLeRi*/
/*This function tells us which metro to get on*/
void headingWhere(int vb)
{
if(vb==65)
printf("\nBoard on a metro heading towards Dwarka Sector 21\n");
else if(vb==67)
printf("\nBoard on a metro heading towards Noida City Center/Vaishali\n");
else if(vb==20)
printf("\nBoard on a metro heading towards Huda City Center\n");
else if(vb==22)
printf("\nBoard on a metro heading towards Samaypur Badli\n");
else if(vb==83)
printf("\nBoard on a metro heading towards Vaishali\n");
else if(vb==71)
printf("\nBoard on a metro heading towards Vaishali\n");
else if(vb==24)
printf("\nBoard on a metro heading towards Huda City Center\n");
else if(vb==26)
printf("\nBoard on a metro heading towards Samaypur Badli\n");
else if(vb==58)
printf("\nBoard on a metro heading towards Dwarka Sector 21\n");
else if(vb==60)
printf("\nBoard on a metro heading towards Noida City Center/Vaishali\n");
else if(vb==132)
printf("\nBoard on a metro heading towards ITO\n");
else if(vb==134)
printf("\nBoard on a metro heading towards Badarpur\n");
}
/*This function is similar to the function 'headingWhere' but it is separated from that function due to some
exceptios*/
void headingWhere2(int vb)
{
if(vb==132)
printf("\nBoard on a metro heading towards Badarpur\n");
else if(vb==130)
printf("\nBoard on a metro heading towards ITO\n");
else if(vb==134)
printf("\nBoard on a metro heading towards Badarpur");
else if(vb==20)
printf("\nBoard on a metro heading towards Samaypur Badli");
else if(vb==20)
printf("\nBoard on a metro heading towards Samaypur Badli");
else if(vb==18)
printf("\nBoard on a metro heading towards Huda City Center");
else if(vb==67)
printf("\nBoard on a metro heading towards Dwarka Sector 21");
else if(vb==69)
printf("\nBoard on a metro heading towards Noida City Center/Vaishali");
}
/*This function tells us which metro to catch on the station i.e The metro which is heading towards station X
or the one which is coming
from the station X*/
```

```
void leftSame(int destColor)
{
if(destColor==Yellow)
printf("Huda City Center");
else if(destColor==Blue)
printf("Dwarka Sect. 21");
else if(destColor==Violet)
printf("ITO");
return;
}
//Similar to leftSame
void rightSame(int destColor)
{
if(destColor==Yellow)
printf("Samaypur Badli\n");
else if(destColor==Blue)
printf("Noida City Center/Vaishali\n");
else if(destColor==Violet)
printf("Badarpur");
return;
}
/*Entered station name is compared with each stations in this function*/
int station(char Name[30])
{
int i;
//yellow line
i=strcasecmp(Name,"Huda City Center");
if(i==0)
return 0;
i=strcasecmp(Name,"Iffco Chowk");
if(i==0)
return 1;
i=strcasecmp(Name,"MG Road");
if(i==0)
return 2;
i=strcasecmp(Name,"Sikanderpur");
if(i==0)
return 3;
i=strcasecmp(Name,"Guru Dronacharya");
if(i==0)
return 4;
i=strcasecmp(Name,"Arjangarh");
if(i==0)
return 5;
i = strcasecmp(Name,"Ghitorni");
if(i ==0)
return 6;
i = strcasecmp(Name,"Sultanpur");
if(i == 0)
return 7;
i = strcasecmp(Name,"Chattarpur");
if(i == 0)
return 8;
i = strcasecmp(Name,"Qutub Minar");
if(i==0)
return 9;
i = strcasecmp(Name,"Saket");
```

```
if(i==0)
return 10;
i =strcasecmp(Name,"Malviya Nagar");
if(i==0)
return 11;
i =strcasecmp(Name,"Hauz Khas");
if(i==0)
return 12;
i = strcasecmp(Name,"Green Park");
if(i==0)
return 13;
i = strcasecmp(Name,"AIIMS");
if(i==0)
return 14;
i= strcasecmp(Name,"INA");
if(i==0)
return 15;
i = strcasecmp(Name,"Jor Bagh");
if(i==0)
return 16;
i = strcasecmp(Name,"Race Course");
if(i==0)
return 17;
i = strcasecmp(Name,"Udyog Bhawan");
if(i==0)
return 18;
i = strcasecmp(Name,"Central Secratariat");
if(i==0)
return 19;
i = strcasecmp(Name,"Patel Chowk");
if(i==0)
return 20;
i = strcasecmp(Name,"Rajiv Chowk");          //Junction
if(i==0)
return 21;
i = strcasecmp(Name,"New Delhi");
if(i==0)
return 22;
i = strcasecmp(Name,"Chawri Bazaar");
if(i==0)
return 23;
i=strcasecmp(Name,"Chandni Chowk");
if(i==0)
return 24;
i=strcasecmp(Name,"Kashmere Gate");
if(i==0)
return 25;
i=strcasecmp(Name,"Civil Lines");
if(i==0)
return 26;
i=strcasecmp(Name,"Vidhan Sabha");
if(i==0)
return 27;
i=strcasecmp(Name,"Vishwavidyalaya");
if(i==0)
return 28;
i=strcasecmp(Name,"GTB Nagar");
```

```
if(i==0)
return 29;
i=strcasecmp(Name,"Model Town");
if(i==0)
return 30;
i=strcasecmp(Name,"Azadpur");
if(i==0)
return 31;
i=strcasecmp(Name,"Adarsh Nagar");
if(i==0)
return 32;
i=strcasecmp(Name,"Jahangirpuri");
if(i==0)
return 33;
i=strcasecmp(Name,"Haiderpur Badli Mor");
if(i==0)
return 34;
i=strcasecmp(Name,"Rohini Sector 18/19");
if(i==0)
return 35;
i=strcasecmp(Name,"Samaypur Badli");
if(i==0)
return 36;

//Blue line
i = strcasecmp(Name,"Dwarka Sect. 21");
if(i==0)
return 38;
i = strcasecmp(Name,"Dwarka Sect. 8");
if(i==0)
return 39;
i = strcasecmp(Name,"Dwarka Sect. 9");
if(i==0)
return 40;
i = strcasecmp(Name,"Dwarka Sect. 10");
if(i==0)
return 41;
i = strcasecmp(Name,"Dwarka Sect. 11");
if(i==0)
return 42;
i = strcasecmp(Name,"Dwarka Sect. 12");
if(i==0)
return 43;
i = strcasecmp(Name,"Dwarka Sect. 13");
if(i==0)
return 44;
i = strcasecmp(Name,"Dwarka Sect. 14");
if(i==0)
return 45;
i = strcasecmp(Name,"Dwarka");
if(i==0)
return 46;
i = strcasecmp(Name,"Dwarka Mor");
if(i==0)
return 47;
i = strcasecmp(Name,"Nawada");
if(i==0)
```

```
return 48;
i = strcasecmp(Name,"Uttam Nagar West");
if(i==0)
return 49;
i = strcasecmp(Name,"Uttam Nagar East");
if(i==0)
return 50;
i = strcasecmp(Name,"Janakpuri West");
if(i==0)
return 51;
i = strcasecmp(Name,"Janakpuri East");
if(i==0)
return 52;
i = strcasecmp(Name,"Tilak Nagar");
if(i==0)
return 53;
i = strcasecmp(Name,"Subhash Nagar");
if(i==0)
return 54;
i = strcasecmp(Name,"Tagore Garden");
if(i==0)
return 55;
i = strcasecmp(Name,"Rajouri Garden");
if(i==0)
return 56;
i = strcasecmp(Name,"Ramesh Nagar");
if(i==0)
return 57;
i = strcasecmp(Name,"Moti Nagar");
if(i==0)
return 58;
i = strcasecmp(Name,"Kirti Nagar");
if(i==0)
return 59;
i = strcasecmp(Name,"Shadipur");
if(i==0)
return 60;
i = strcasecmp(Name,"Patel Nagar");
if(i==0)
return 61;
i = strcasecmp(Name,"Rajendra Place");
if(i==0)
return 62;
i = strcasecmp(Name,"Karol Bagh");
if(i==0)
return 63;
i = strcasecmp(Name,"Jhandewalan");
if(i==0)
return 64;
i = strcasecmp(Name,"Ramakrishna Ashram Marg");
if(i==0)
return 65;
i = strcasecmp(Name,"Rajiv Chowk");   //junction
if(i==0)
return 66;
i = strcasecmp(Name,"Barakhamba Road");
if(i==0)
```

```
return 67;
i = strcasecmp(Name,"Mandi House");
if(i==0)
return 68;
i = strcasecmp(Name,"Pragati Maidan");
if(i==0)
return 69;
i = strcasecmp(Name,"Indraprastha");
if(i==0)
return 70;
i = strcasecmp(Name,"Yamuna Bank");
if(i==0)
return 71;
i=strcasecmp(Name,"Akshardham");
if(i==0)
return 73;
i=strcasecmp(Name,"Mayur Vihar I");
if(i==0)
return 74;
i=strcasecmp(Name,"Mayur Vihar Ext.");
if(i==0)
return 75;
i=strcasecmp(Name,"New Ashok Nagar");
if(i==0)
return 76;
i=strcasecmp(Name,"Noida Sect. 15");
if(i==0)
return 77;
i=strcasecmp(Name,"Noida Sect. 16");
if(i==0)
return 78;
i=strcasecmp(Name,"Noida Sect. 18");
if(i==0)
return 79;
i=strcasecmp(Name,"Botanical Garden");
if(i==0)
return 80;
i=strcasecmp(Name,"Golf Course");
if(i==0)
return 81;
i=strcasecmp(Name,"Noida City Center");
if(i==0)
return 82;
i=strcasecmp(Name,"Lakshmi Nagar");
if(i==0)
return 83;
i=strcasecmp(Name,"Nirman Vihar");
if(i==0)
return 84;
i=strcasecmp(Name,"Preet Vihar");
if(i==0)
return 85;
i=strcasecmp(Name,"Karkarduma");
if(i==0)
return 86;
i=strcasecmp(Name,"Anand Vihar ISBT");
if(i==0)
```

```
return 87;
i=strcasecmp(Name,"Kaushambi");
if(i==0)
return 88;
i=strcasecmp(Name,"Vaishali");
if(i==0)
return 89;
//violet line
i=strcasecmp(Name,"ITO");
if(i==0)
return 130;
i=strcasecmp(Name,"Mandi House");
if(i==0)
return 131;
i=strcasecmp(Name,"Janpath");
if(i==0)
return 132;
i=strcasecmp(Name,"Central Secratariat");
if(i==0)
return 133;
i=strcasecmp(Name,"Khan Market");
if(i==0)
return 134;
i=strcasecmp(Name,"JLN Stadium");
if(i==0)
return 135;
i=strcasecmp(Name,"Jangpura");
if(i==0)
return 136;
i=strcasecmp(Name,"Lajpat Nagar");
if(i==0)
return 137;
i=strcasecmp(Name,"Moolchand");
if(i==0)
return 138;
i=strcasecmp(Name,"Kailash Colony");
if(i==0)
return 139;
i=strcasecmp(Name,"Nehru Place");
if(i==0)
return 140;
i=strcasecmp(Name,"Kalkaji Mandir");
if(i==0)
return 141;
i=strcasecmp(Name,"Govind Puri");
if(i==0)
return 142;
i=strcasecmp(Name,"Okhla");
if(i==0)
return 143;
i=strcasecmp(Name,"Jasola - Apollo");
if(i==0)
return 144;
i=strcasecmp(Name,"Sarita Vihar");
if(i==0)
return 145;
i=strcasecmp(Name,"Mohan Estate");
```

```
if(i==0)
return 146;
i=strcasecmp(Name,"Tughlakabad");
if(i==0)
return 147;
i=strcasecmp(Name,"Badarpur Border");
if(i==0)
return 148;
i=strcasecmp(Name,"Sarai");
if(i==0)
return 149;
i=strcasecmp(Name,"NHPC Chowk");
if(i==0)
return 150;
i=strcasecmp(Name,"Mewala Maharajpur");
if(i==0)
return 151;
i=strcasecmp(Name,"Sector 28");
if(i==0)
return 152;
i=strcasecmp(Name,"Badkal Mor");
if(i==0)
return 153;
i=strcasecmp(Name,"Old Faridabad");
if(i==0)
return 154;
i=strcasecmp(Name,"Neelam Chand Ajronda");
if(i==0)
return 155;
i=strcasecmp(Name,"Bata Chowk");
if(i==0)
return 156;
i=strcasecmp(Name,"Escorts Mujesar");
if(i==0)
return 157;
else
return -50;
}
void numToName(int t)
{
switch(t)
{
//Yellow Line
case 0 :
printf("\nHuda City Center->");
break;
case 1 :
printf("\nIffco Chowk->");
break;
case 2 :
printf("\nMG Road->");
break;
case 3 :
printf("\nSikanderpur->");
break;
case 4 :
printf("\nGuru Dronacharya->");
```

```
break;
case 5 :
printf("\nArjangarh->");
break;
case 6 :
printf("\nGhitorni->");
break;
case 7 :
printf("\nSultanpur->");
break;
case 8 :
printf("\nChattarpur->");
break;
case 9 :
printf("\nQutub Minar->");
break;
case 10 :
printf("\nSaket->");
break;
case 11 :
printf("\nMalviya Nagar->");
break;
case 12 :
printf("\nHauz Khas->");
break;
case 13:
printf("\nGreen Park->");
break;
case 14 :
printf("\nAIIMS->");
break;
case 15 :
printf("\nINA->");
break;
case 16 :
printf("\nJor Bagh->");
break;
case 17 :
printf("\nRace Course->");
break;
case 18 :
printf("\nUdyog Bhawan->");
break;
case 19:
printf("\nCentral Secrateriat->");
break;
case 20 :
printf("\nPatel Chowk->");
break;
case 21 :
printf("\nRajiv Chowk->");
break;
case 22 :
printf("\nNew Delhi->");
break;
case 23 :
printf("\nChawri Bazaar->");
```

```
break;
case 24 :
printf("\nChandni Chowk->");
break;
case 25 :
printf("\nKashmere Gate->");
break;
case 26 :
printf("\nCivil Lines->");
break;
case 27 :
printf("\nVidhan Sabha->");
break;
case 28 :
printf("\nVishwavidyalaya->");
break;
case 29 :
printf("\nGTB Nagar->");
break;
case 30 :
printf("\nModel Town->");
break;
case 31 :
printf("\nAzadpur->");
break;
case 32 :
printf("\nAdarsh Nagar->");
break;
case 33 :
printf("\nJahangirpuri->");
break;
case 34 :
printf("\nHaiderpur Badli Mor->");
break;
case 35 :
printf("\nRohini Sector 18,19->");
break;
case 36 :
printf("\nSamaypur Badli->");
break;
//Blue Line
case 38 :
printf("\nDwarka Sector 21->");
break;
case 39 :
printf("\nDwarka Sector 8->");
break;
case 40 :
printf("\nDwarka Sector 9->");
break;
case 41 :
printf("\nDwarka Sector 10->");
break;
case 42 :
printf("\nDwarka Sector 11->");
break;
case 43 :
```

```
printf("\nDwarka Sector 12->");
break;
case 44 :
printf("\nDwarka Sector 13->");
break;
case 45 :
printf("\nDwarka Sector 14->");
break;
case 46 :
printf("\nDwarka->");
break;
case 47 :
printf("\nDwarka Mor->");
break;
case 48 :
printf("\nNawada->");
break;
case 49 :
printf("\nUttam Nagar West->");
break;
case 50 :
printf("\nUttam Nagar East->");
break;
case 51 :
printf("\nJanakpuri West");
break;
case 52 :
printf("\nJanakpuri East");
break;
case 53 :
printf("\nTilak Nagar->");
break;
case 54 :
printf("\nSubhas Nagar->");
break;
case 55 :
printf("\nTagore Garden->");
break;
case 56 :
printf("\nRajouri Garden->");
break;
case 57 :
printf("\nRamesh Nagar->");
break;
case 58 :
printf("\nMoti Nagar->");
break;
case 59 :
printf("\nKirti Nagar->");
break;
case 60 :
printf("\nShadipur->");
break;
case 61 :
printf("\nPatel Nagar->");
break;
case 62 :
```

```
printf("\nRajendra Place->");
break;
case 63 :
printf("\nKarol Bagh->");
break;
case 64 :
printf("\nJhandewalan->");
break;
case 65 :
printf("\nRamakrishna Ashram Marg->");
break;
case 66 :
printf("\nRajiv Chowk->");
break;
case 67 :
printf("\nBarakhamba Road->");
break;
case 68 :
printf("\nMandi House->");
break;
case 69 :
printf("\nPragati Maidan->");
break;
case 70 :
printf("\nIndraprastha->");
break;
case 71 :
printf("\nYamuna Bank->");
break;
case 73 :
printf("\nAkshardham->");
break;
case 74 :
printf("\nMayur Vihar I->");
break;
case 75 :
printf("\nMayur Vihar Extn.->");
break;
case 76 :
printf("\nNew Ashok Nagar->");
break;
case 77 :
printf("\nNoida Sect. 15->");
break;
case 78 :
printf("\nNoida Sect. 16->");
break;
case 79 :
printf("\nNoida Sect. 18->");
break;
case 80 :
printf("\nBotanical Garden->");
break;
case 81 :
printf("\nGolf Cource->");
break;
case 82 :
```

```
printf("\nCity Center->");
break;
case 83 :
printf("\nLaxmi Nagar->");
break;
case 84 :
printf("\nNirman Vihar->");
break;
case 85 :
printf("\nPreet Vihar->");
break;
case 86 :
printf("\nKarkarduma->");
break;
case 87 :
printf("\nAnand Vihar ISBT->");
break;
case 88 :
printf("\nKaushambi->");
break;
case 89 :
printf("\nVaishali->");
break;
case 130 :
printf("\nITO->");
break;
case 131 :
printf("\nMandi House->");
break;
case 132 :
printf("\nJanpath->");
break;
case 133 :
printf("\nCentral Secratariat->");
break;
case 134 :
printf("\nKhan Market->");
break;
case 135 :
printf("\nJLN Stadium->");
break;
case 136 :
printf("\nJangpura->");
break;
case 137 :
printf("\nLajpat Nagar->");
break;
case 138 :
printf("\nMoolchand->");
break;
case 139 :
printf("\nKailash Colony->");
break;
case 140 :
printf("\nNehru Place->");
break;
case 141 :
```

```
printf("\nKalkaji Mandir->");
break;
case 142 :
printf("\nGovind Puri->");
break;
case 143 :
printf("\nOkhla->");
break;
case 144 :
printf("\nJasola - Appolo->");
break;
case 145 :
printf("\nSarita Vihar->");
break;
case 146 :
printf("\nMohan Estate->");
break;
case 147 :
printf("\nTughlakabad->");
break;
case 148 :
printf("\nBadarpur Border->");
break;
case 149 :
printf("\nSarai->");
break;
case 150 :
printf("\nNHPC Chowk->");
break;
case 151 :
printf("\nMewala Maharajpur->");
break;
case 152 :
printf("\nSector 28->");
break;
case 153 :
printf("\nBadkal Mor->");
break;
case 154 :
printf("\nOld Faridabad->");
break;
case 155 :
printf("\nNeelam Chowk Ajronda->");
break;
case 156 :
printf("\nBata Chowk->");
break;
case 157 :
printf("\nEscorts Mujesar->");
break;
return;
}
}
void push3(int data)
{
if(top==MAXMATCH-1)
{
```

```
printf("\nStack Overflow");
exit(1);
}
top=top+1;
Match3[top]=data;
return;
}
int pop3()
{
int data;
if(top==-1)
{
printf("\nStack Underflow\n");
   return 0;
}
data=Match3[top];
top=top-1;
return data;
}
int colorDecider(int Number)
{
int Color;
if(Number>=0 && Number <=37)
Color = Yellow;
else if(Number>=38 && Number <=89)
Color = Blue;
else if(Number>=130 && Number <=157)
Color = Violet;
return Color;
}
void createGraph()//This function connects stations according to Delhi Metro routes
{
adj[0][1]=4;
adj[1][0]=4;
adj[1][2]=4;
adj[2][1]=4;
adj[2][3]=4;
adj[3][2]=4;
adj[3][4]=4;
adj[4][3]=4;
adj[4][5]=4;
adj[5][4]=4;
adj[5][6]=4;
adj[6][5]=4;
adj[6][7]=4;
adj[7][6]=4;
adj[7][8]=4;
adj[8][7]=4;
adj[8][9]=4;
adj[9][8]=4;
adj[9][10]=4;
adj[10][9]=4;
adj[10][11]=4;
adj[11][10]=4;
adj[11][12]=4;
adj[12][11]=4;
adj[12][13]=4;
```

```
adj[13][12]=4;
adj[13][14]=4;
adj[14][13]=4;
adj[14][15]=4;
adj[15][14]=4;
adj[15][16]=4;
adj[16][15]=4;
adj[16][17]=4;
adj[17][16]=4;
adj[17][18]=4;
adj[18][17]=4;
adj[18][19]=4;
adj[19][18]=4;
adj[19][20]=4;
adj[19][133]=5;       //Yellow to Violet
adj[20][19]=4;
adj[20][21]=4;
adj[21][66]=5;         //Yellow to Blue
adj[21][20]=4;
adj[21][22]=4;
adj[22][21]=4;
adj[22][23]=4;
adj[23][22]=4;
adj[23][24]=4;
adj[24][23]=4;
adj[24][25]=4;
adj[25][24]=4;
adj[25][26]=4;
adj[26][25]=4;
adj[26][27]=4;
adj[27][26]=4;
adj[27][28]=4;
adj[28][27]=4;
adj[28][29]=4;
adj[29][28]=4;
adj[29][30]=4;
adj[30][29]=4;
adj[30][31]=4;
adj[31][30]=4;
adj[31][32]=4;
adj[32][31]=4;
adj[32][33]=4;
adj[33][32]=4;
adj[33][34]=4;
adj[34][33]=4;
adj[34][35]=4;
adj[35][34]=4;
adj[35][36]=4;
adj[36][35]=4;
adj[38][39]=4;
adj[39][38]=4;
adj[39][40]=4;
adj[40][39]=4;
adj[40][41]=4;
adj[41][40]=4;
adj[41][42]=4;
adj[42][41]=4;
```

```
adj[42][43]=4;
adj[43][42]=4;
adj[43][44]=4;
adj[44][43]=4;
adj[44][45]=4;
adj[45][44]=4;
adj[45][46]=4;
adj[46][45]=4;
adj[46][47]=4;
adj[47][46]=4;
adj[47][48]=4;
adj[48][47]=4;
adj[48][49]=4;
adj[49][48]=4;
adj[49][50]=4;
adj[50][49]=4;
adj[50][51]=4;
adj[51][50]=4;
adj[51][52]=4;
adj[52][51]=4;
adj[52][53]=4;
adj[53][52]=4;
adj[53][54]=4;
adj[54][53]=4;
adj[54][55]=4;
adj[55][54]=4;
adj[55][56]=4;
adj[56][55]=4;
adj[56][57]=4;
adj[57][56]=4;
adj[57][58]=4;
adj[58][57]=4;
adj[58][59]=4;
adj[59][58]=4;
adj[59][60]=4;
adj[60][59]=4;
adj[60][61]=4;
adj[61][60]=4;
adj[61][62]=4;
adj[62][61]=4;
adj[62][63]=4;
adj[63][62]=4;
adj[63][64]=4;
adj[64][63]=4;
adj[64][65]=4;
adj[65][64]=4;
adj[65][66]=4;
adj[66][65]=4;
adj[66][67]=4;
adj[66][21]=5;      //Blue to Yellow
adj[67][66]=4;
adj[67][68]=4;
adj[68][67]=4;
adj[68][69]=4;
adj[68][131]=5;      //Blue to violet
adj[69][68]=4;
adj[69][70]=4;
```

```
adj[70][69]=4;
adj[70][71]=4;
adj[71][70]=4;   //Yamuna Bank
adj[71][73]=4;   //Towards Noida
adj[71][83]=4;   //Towards Vaishali
adj[73][71]=4;   //Akshardham to Yamuna Bank
adj[73][74]=4;
adj[74][73]=4;
adj[74][75]=4;
adj[75][74]=4;
adj[75][76]=4;
adj[76][75]=4;
adj[76][77]=4;
adj[77][76]=4;
adj[77][78]=4;
adj[78][77]=4;
adj[78][79]=4;
adj[79][78]=4;
adj[79][80]=4;
adj[80][79]=4;
adj[80][81]=4;
adj[81][80]=4;
adj[81][82]=4;
adj[82][81]=4;  //City Center
adj[83][71]=4;  //Laxmi Nagar to Yamuna Bank
adj[83][84]=4;
adj[84][83]=4;
adj[84][85]=4;
adj[85][84]=4;
adj[85][86]=4;
adj[86][85]=4;
adj[86][87]=4;
adj[87][86]=4;
adj[87][88]=4;
adj[88][87]=4;
adj[88][89]=4;
adj[89][88]=4;  //Vaishali
adj[130][131]=4;
adj[131][130]=4;
adj[131][132]=4;
adj[131][68]=5;  //Violet to blue line
adj[132][131]=4;
adj[132][133]=4;
adj[133][132]=4;
adj[133][134]=4;
adj[133][19]=5;   //Violet to Yellow
adj[134][133]=4;
adj[134][135]=4;
adj[135][134]=4;
adj[135][136]=4;
adj[136][135]=4;
adj[136][137]=4;
adj[137][136]=4;
adj[137][138]=4;
adj[138][137]=4;
adj[138][139]=4;
adj[139][138]=4;
```

```
adj[139][140]=4;
adj[140][139]=4;
adj[140][141]=4;
adj[141][140]=4;
adj[141][142]=4;
adj[142][141]=4;
adj[142][143]=4;
adj[143][142]=4;
adj[143][144]=4;
adj[144][143]=4;
adj[144][145]=4;
adj[145][144]=4;
adj[145][146]=4;
adj[146][145]=4;
adj[146][147]=4;
adj[147][146]=4;
adj[147][148]=4;
adj[148][147]=4;
adj[148][149]=4;
adj[149][148]=4;
adj[149][150]=4;
adj[150][149]=4;
adj[150][151]=4;
adj[151][150]=4;
adj[151][152]=4;
adj[152][151]=4;
adj[152][153]=4;
adj[153][152]=4;
adj[153][154]=4;
adj[154][153]=4;
adj[154][155]=4;
adj[155][154]=4;
adj[155][156]=4;
adj[156][155]=4;
adj[156][157]=4;
adj[157][156]=4;
}
```