ABB-1 MAP

Project Report

DETAILS OF THE TEAM

(4 Members)

1. *RHYTHM SRIVASTAVA 21103234 - B11*
2. *SRISHTI* GARG 21103227 - *B11*
3. *KAMAL GARG 21103231 - B11*
4. *HIMANSHU DIXIT 21103262 - B11*

Submitted To :-

Ms. Sherry Garg

ACKNOWLEDGEMENT

We have taken a lot of effort into this project. However, completing this project would not have been possible without the support and guidance of a lot of individuals. We would like to extend our sincere thanks to all of them.

We are highly indebted to Ms. Sherry Garg Mam for her guidance and supervision. We would like to thank her for providing the necessary information and resources for this project.

We would like to express our gratitude towards our parents and our friends for their kind co-operation and encouragement which help us a lot in completing this project.

Our thanks and appreciations also go to our colleagues in developing the project. Thank you to all the people who have willingly helped us out with their abilities.

ABSTRACT OF THE PROJECT

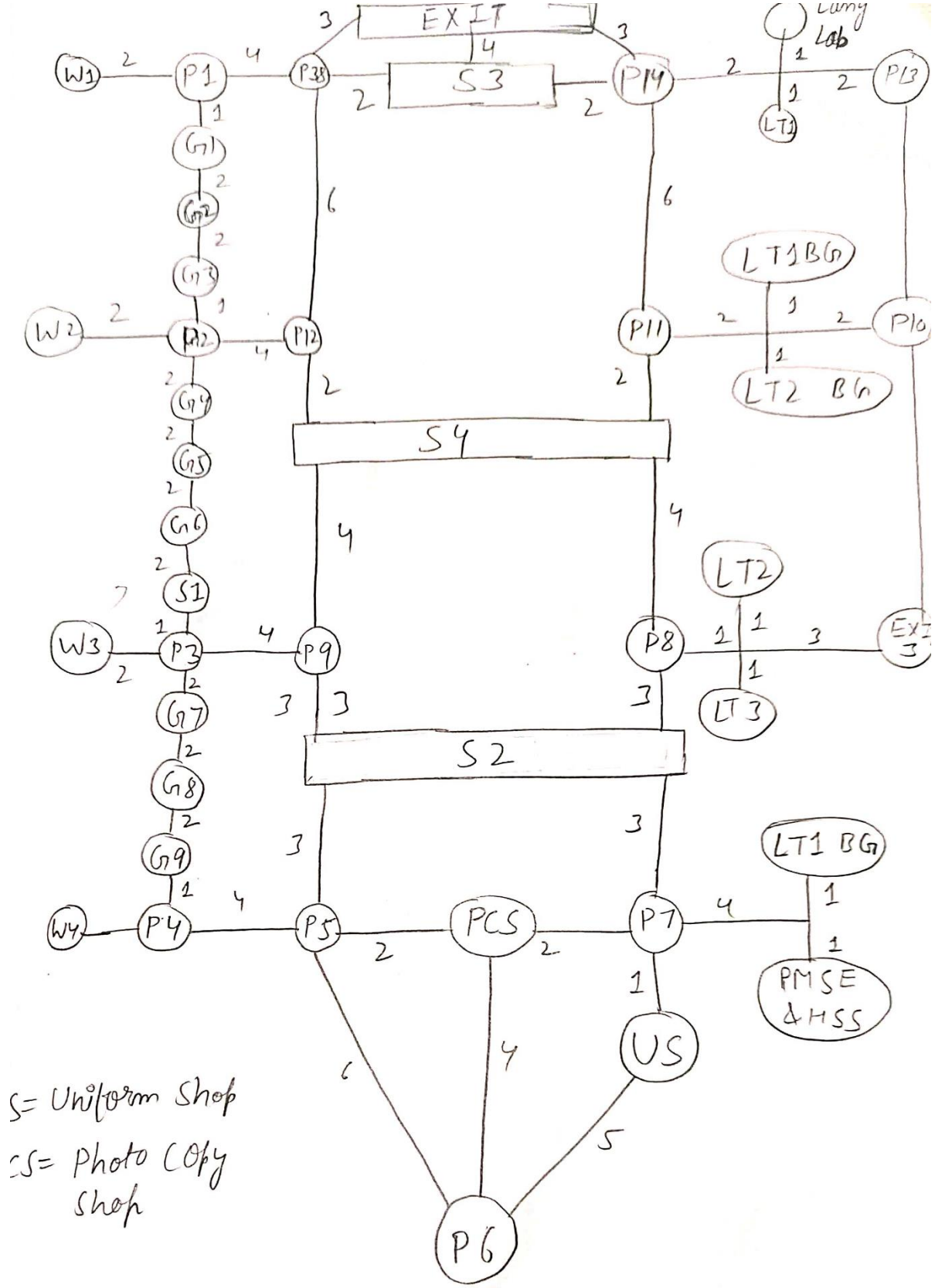
This C++ project made by us is a saviour to all the freshers of our college Jaypee Institute Of Information Technology, Noida, Sector-62. Searching for classes becomes a hustle for them in between the change overs due to which they get late for their classes. So we have attempted to listen to their cries by making a project which takes the input of the class at which they are currently present (source) and the class to which they want to move to (destination) and they will get the detailed shortest path as a series of identifiable points to reach the class they wanted to move to as the output. It is as simple as that. This program is a map of academic building ABB-1 of JIIT. It covers all the three floors of this building along with the passages, staircases, wing entries and exits. ABB-1 map has been made with the objective of making the system reliable,easier,fast and more informative. This C++ code mainly makes use of graphs along with a famous shortest path algorithm Dijsktra's.

TOPICS USED IN THIS PROJECT

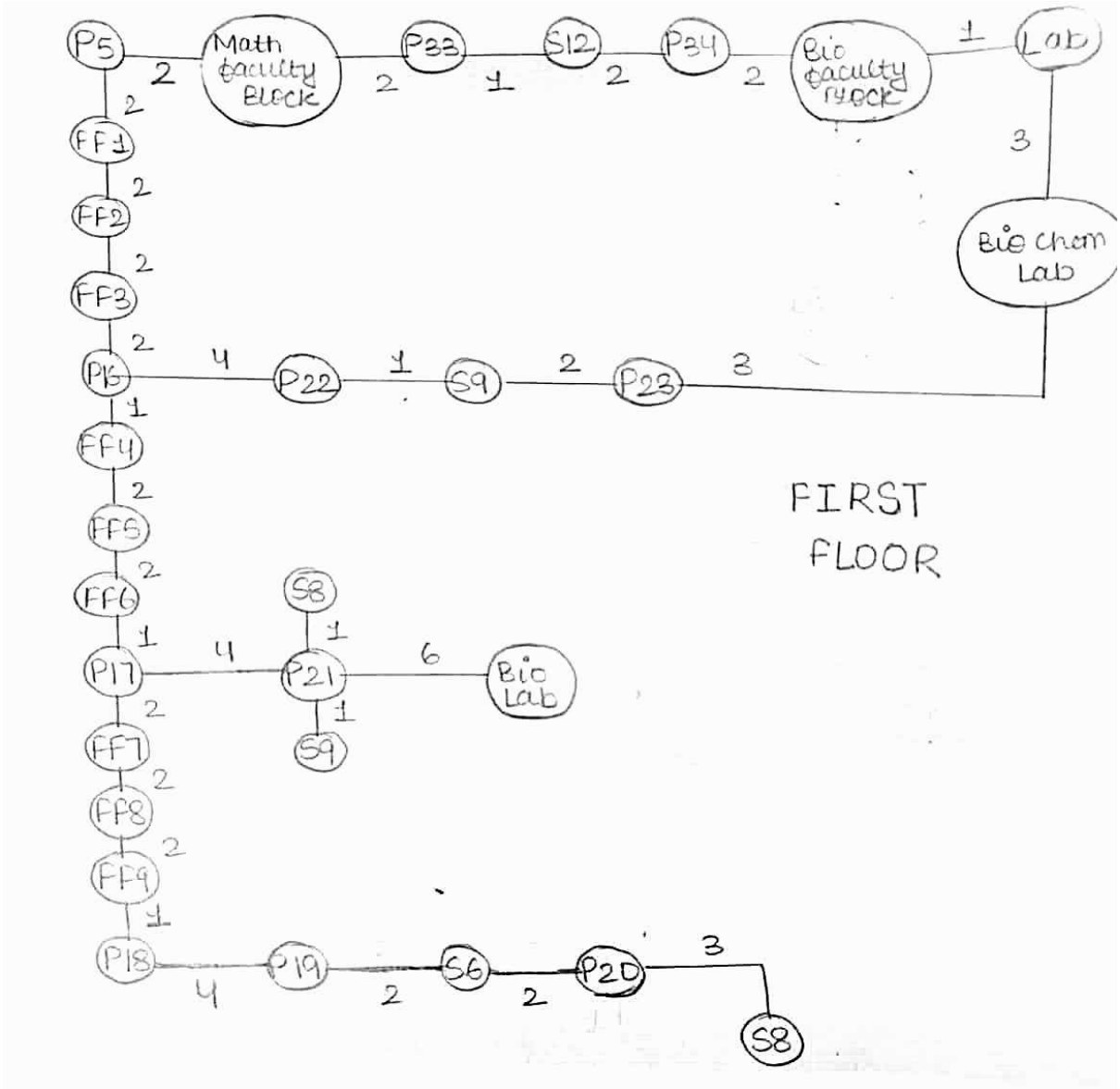
- Graphs
- Classes
- If Else statements
- Functions
- Strings
- Array
- Switch case
- Loops
- Dijkstra’s Algorithm
- template

REFERENCE MAP

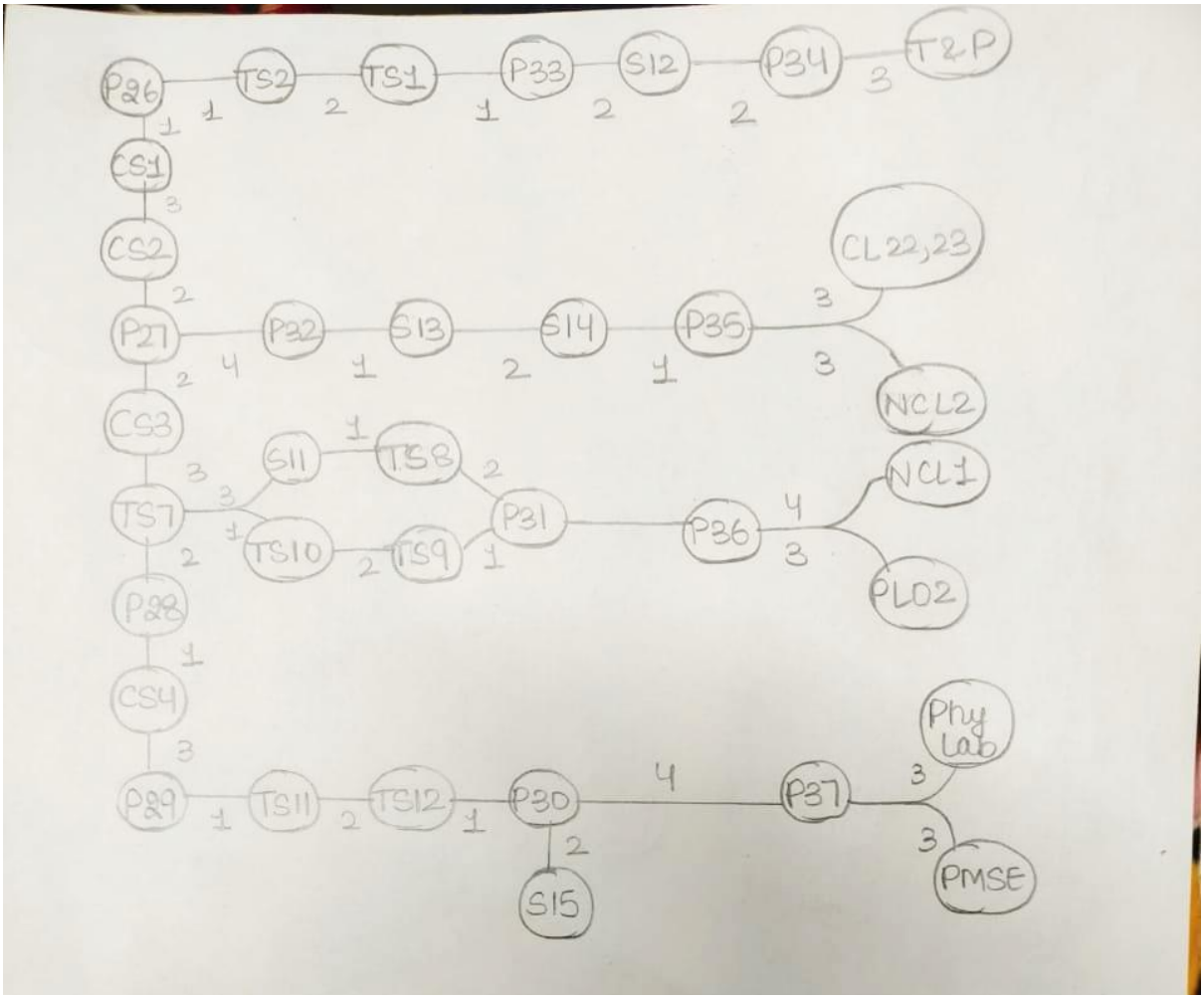
Ground FLOOR:



FIRST FLOOR:


P5
Math faculty Block
P33
S12
P34
Bio faculty block
Lab
Bio chem Lab
FF1
FF2
FF3
P16
P22
S9
P23
FF4
FF5
FF6
S8
P17
P21
Bio Lab
S9
FF7
FF8
FF9
P18
P19
S6
P20
S8
FIRST FLOOR

SECOND FLOOR:



SOURCE CODE

```
//-------------------------------------------------------------------------------------------------
// header files used in project
//-------------------------------------------------------------------------------------------------
#include <iostream>
#include <time.h>
#include <conio.h>
#include <climits>
#include <cstring>

using namespace std;

#define v 112
int arr[v][v]{0};
void help(int u, int x, int d)
{
    arr[u][x] = d;
    arr[x][u] = d;
}

template <typename T>
class vector
{
    T *arr;

    // capacity of the vector
    int capacity;

    // current is the number of elements
    int current;

public:
    // constructor
    vector()
    {
        arr = new T[1];
        capacity = 1;
        current = 0;
    }

    vector(int c, int val = 0)
    {
        arr = new T[c];
        for (int i = 0; i < c; i++)
        {
            arr[i] = val;
        }
        capacity = c;
        current = 0;
    }

    // destructor
    ~vector()
    {
        delete[] arr;
```

```
}

void push(T data)
{

    /* if the number of elements is equal to the
    capacity, that means we don't have space to
    accommodate more elements. We need to double the
    capacity */
    if (current == capacity)
    {
        T *temp = new T[2 * capacity];

        // copying old array elements to new array
        for (int i = 0; i < capacity; i++)
        {
            temp[i] = arr[i];
        }

        // deleting previous array
        delete[] arr;
        capacity *= 2;
        arr = temp;
    }

    // Inserting data
    arr[current] = data;
    current++;
}

void push(T data, int index)
{
    if (index == capacity)
        push(data);
    else
        arr[index] = data;
}

// function to extract element at any index
T get(int index)
{
    if (index < current)
        return arr[index];
}

// function to delete last element
void pop()
{
    current--;
}

// function to get size of the vector
int size()
{
    return current;
}

// function to get capacity of the vector
int getcapacity()
{
    return capacity;
}
```

```
    // function to print array elements
    void print()
    {
        for (int i = 0; i < current; i++)
        {
            cout << arr[i] << " ";
        }
        cout << endl;
    }

    T &operator[](int i)
    {
        return arr[i];
    }
};

vector<int> dist(v, INT_MAX);
vector<int> parents(v);
string names[] = {
    "W1",
    "P1",
    "G1",
    "G2",
    "G3",
    "P2",
    "G4",
    "G5",
    "G6",
    "S1",
    "P3",
    "G7",
    "G8",
    "G9",
    "P4",
    "P5",
    "P6",
    "US",
    "P7",
    "Photo Copy Shop",
    "PMSE & HSS",
    "LT3 BG",
    "S2",
    "EXIT 3",
    "LT3",
    "LT2",
    "P8",
    "P9",
    "P10",
    "LT2 BG",
    "P11",
    "LT1 BG",
    "P12",
    "LT1",
    "P13",
    "LANG LAB",
    "P14",
    "S3",
    "EXIT",
    "S4",
    "P15",
    "FF1",
```

```
    "FF2",
    "FF3",
    "P16",
    "FF4",
    "FF5",
    "FF6",
    "S5",
    "P17",
    "FF7",
    "FF8",
    "FF9",
    "P18",
    "P19",
    "S6",
    "P20",
    "S7",
    "BIO LAB",
    "P21",
    "S8",
    "BIO LAB",
    "P22",
    "S9",
    "P23",
    "BIO CHEM LAB",
    "BIO CHEM LAB",
    "BIO FACULTY",
    "P24",
    "S10",
    "P25",
    "MATH FACULTY",
    "P26",
    "CS1",
    "CS2",
    "P27",
    "CS3",
    "TS7",
    "P28",
    "CS4",
    "P29",
    "TS11",
    "TS12",
    "P30",
    "TS10",
    "TS9",
    "P31",
    "S11",
    "TS8",
    "P32",
    "TS2",
    "TS1",
    "P33",
    "S12",
    "P34",
    "TnP",
    "S13",
    "S14",
    "P35",
    "CL22 AND CL23",
    "NCL1",
    "NCL2",
    "P36",
    "PL02",
```

```
    "PL01",
    "P37",
    "PMSE",
    "S15",
    "P38",
    "W2",
    "W3",
    "W4",
};
static int a = 0;
void printPath(int currentVertex, vector<int> &parents)
{
    // Base case : Source node has
    // been processed
    if (currentVertex == -1)
    {
        return;
    }

    printPath(parents[currentVertex], parents);
    if (a++ > 0)
    {

        cout << "--->";
    }

    cout << names[currentVertex] << " ";
}

// A utility function to print
// the constructed distances
// array and shortest paths

int printSolution(int startVertex, vector<int> &distances,
                  vector<int> &parents, int d, int choice)
{

    if (choice == 1)
    {
        cout << "\n\n\t\t\tSource\t\tDestination\t\tDistance";
        cout << "\n\n\t\t\t-------------------------------------------------";
    }
    else if (choice == 2)
    {
        cout << "\n\n\t\t\tSource\t\tDestination\t\tPath";
        cout << "\n\n\t\t\t-----------------------------------------";
    }
    else if (choice == 3)
    {
        cout << "\n\n\t\t\tSource\t\tDestination\t\tDistance\t\tPath";
        cout << "\n\n\t\t\t-----------------------------------------------------------------------
";
    }

    for (int vertexIndex = 0; vertexIndex < v;
         vertexIndex++)
    {
        if (vertexIndex != startVertex && vertexIndex == d)
        {
            cout << "\n\n\t\t\t" << names[startVertex] << " \t\t ";
            cout << names[vertexIndex] << " \t\t\t   ";
            return distances[vertexIndex];
```

```
        }
    }
    return 0;
}

int dijkstra(int graph[][v], int s, int d, int choice)
{

    dist[s] = 0;

    parents[s] = -1;
    vector<bool> fin(v, 0);

    for (int j = 0; j < v - 1; j++)
    {
        int u = -1, i;
        for (i = 0; i < v; i++)
        {
            if (!fin[i] && (u == -1 || dist[i] < dist[u]))
            {
                u = i;
            }
        }

        fin[u] = 1;

        for (int x = 0; x < v; x++)
        {
            if (graph[u][x] != 0 && fin[x] == 0)
            {
                dist[x] = min(dist[x], dist[u] + graph[u][x]);
                if (dist[x] == dist[u] + graph[u][x])
                    parents[x] = u;
            }
        }
    }

    return printSolution(s, dist, parents, d, choice);
}

//-------------------------------------------------------------------------------------------------
// function to print names of team members
//-------------------------------------------------------------------------------------------------

void printMessageCenter(const char *message)
{
    int len = 0;
    int pos = 0;
    // calculate how many space need to print

    len = (78 - strlen(message)) / 2;

    for (pos = 0; pos < len; pos++)
    {
        // print space
        cout << " ";
    }
    // print message
    cout << message << endl;
    cout << "\nDate: " << __DATE__ << "\t\t\t\t\t     Time: " << __TIME__;
```

```
    cout << "\n\n\t\t       1. HIMANSHU DIXIT   - B11 - 21103262";
    cout << "\n\n\t\t       2. KAMAL GARG       - B11 - 21103231";
    cout << "\n\n\t\t       3. SRISHTI GARG     - B11 - 21103227";
    cout << "\n\n\t\t       4. RHYTHM SRIVASTAV - B11 - 21103234\n";
}

//------------------------------------------------------------------------------------------------
// function to heading of page
//------------------------------------------------------------------------------------------------

void printMessageCenter2(const char *message)
{
    int len = 0;
    int pos = 0;
    // calculate how many space need to print

    len = (78 - strlen(message)) / 2;

    for (pos = 0; pos < len; pos++)
    {
        // print space
        cout << " ";
    }
    // print message
    cout << message << endl;
}

//------------------------------------------------------------------------------------------------
// function of small loading page
//------------------------------------------------------------------------------------------------

void loadingpage()
{
    cout << "\t\t\tPLEASE WAIT\n";
    cout << "\t\t\tSYSTEM IS LOADING\n";
    cout << "\n\nPress any key to start now.....";
    getch();
}

//------------------------------------------------------------------------------------------------
// function to display our project name
//------------------------------------------------------------------------------------------------

void headMessage(const char *message)
{
    system("cls");
    cout << "\n###############################################################################";
    cout << "\n############                                                   ############";
    cout << "\n############      THE MAP OF ABB-1 OF JIIT (PROJECT IN C++)    ############";
    cout << "\n############                                                   ############";
    cout << "\n###############################################################################";
    cout << "\n--------------------------------------------------------------------------------\n";
    printMessageCenter(message);
    cout << "\n--------------------------------------------------------------------------------
\n\n";
}
```

```
//-----------------------------------------------------------------------------------------------
-----------
// function to display welcome message
//-----------------------------------------------------------------------------------------------
-----------

void Welcome_Msg()
{
    headMessage("CREATED BY :-");
    loadingpage();
    cout << "\n\n\n\n\n";
    cout << "\n\t    **-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**\n";
    cout << "\n\t          =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=";
    cout << "\n\t          =                 WELCOME                   =";
    cout << "\n\t          =                   TO                      =";
    cout << "\n\t          =                 ABB - 1                   =";
    cout << "\n\t          =                   MAP                     =";
    cout << "\n\t          =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=";
    cout << "\n\n\t    **-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**-**\n";
    cout << "\n\n Enter any key to continue.....";
    getch();
}

//-----------------------------------------------------------------------------------------------
-----------
// function to display MENU
//-----------------------------------------------------------------------------------------------
-----------

void Menu()
{
    int choice = 0;
    int i, j, source, destination;
    string s, d;

    help(0, 1, 2);
    help(1, 0, 2);
    help(1, 2, 1);
    help(2, 1, 1);
    help(2, 3, 2);
    help(3, 2, 2);
    help(3, 4, 2);
    help(4, 3, 2);
    help(4, 5, 1);
    help(5, 4, 1);
    help(5, 6, 2);
    help(6, 5, 2);
    help(5, 32, 4);
    help(32, 5, 4);
    help(6, 7, 2);
    help(7, 6, 2);
    help(7, 8, 2);
    help(8, 7, 2);
    help(8, 9, 2);
    help(9, 8, 2);
    help(10, 9, 1);
    help(10, 11, 2);
    help(11, 10, 2);
    help(10, 27, 4);
    help(27, 10, 4);
    help(11, 12, 2);
```

```
    help(12, 11, 2);
    help(12, 13, 2);
    help(13, 12, 2);
    help(13, 14, 1);
    help(14, 13, 1);
    help(14, 15, 4);
    help(15, 16, 6);
    help(16, 15, 6);
    help(15, 19, 2);
    help(19, 15, 2);
    help(22, 15, 3);
    help(15, 22, 3);
    help(16, 19, 4);
    help(19, 16, 4);
    help(16, 17, 5);
    help(17, 16, 5);
    help(17, 18, 1);
    help(18, 19, 2);
    help(19, 18, 2);
    help(18, 20, 5);
    help(20, 18, 5);
    help(18, 21, 5);
    help(21, 18, 5);
    help(18, 22, 3);
    help(22, 18, 3);
    help(22, 26, 3);
    help(26, 22, 3);
    help(22, 27, 3);
    help(27, 22, 3);
    help(26, 24, 2);
    help(24, 26, 2);
    help(24, 25, 2);
    help(25, 24, 2);
    help(25, 23, 4);
    help(23, 25, 4);
    help(24, 23, 4);
    help(23, 24, 4);
    help(26, 39, 4);
    help(39, 26, 4);
    help(27, 39, 4);
    help(39, 27, 4);
    help(23, 28, 8);
    help(28, 23, 8);
    help(39, 32, 2);
    help(32, 32, 2);
    help(39, 30, 2);
    help(30, 39, 2);
    help(30, 31, 3);
    help(31, 30, 3);
    help(30, 29, 3);
    help(29, 30, 3);
    help(31, 28, 3);
    help(28, 31, 3);
    help(29, 28, 3);
    help(28, 29, 3);
    help(32, 37, 6);
    help(37, 32, 6);
    help(30, 36, 6);
    help(36, 30, 6);
    help(36, 37, 2);
    help(37, 36, 2);
    help(37, 38, 4);
```

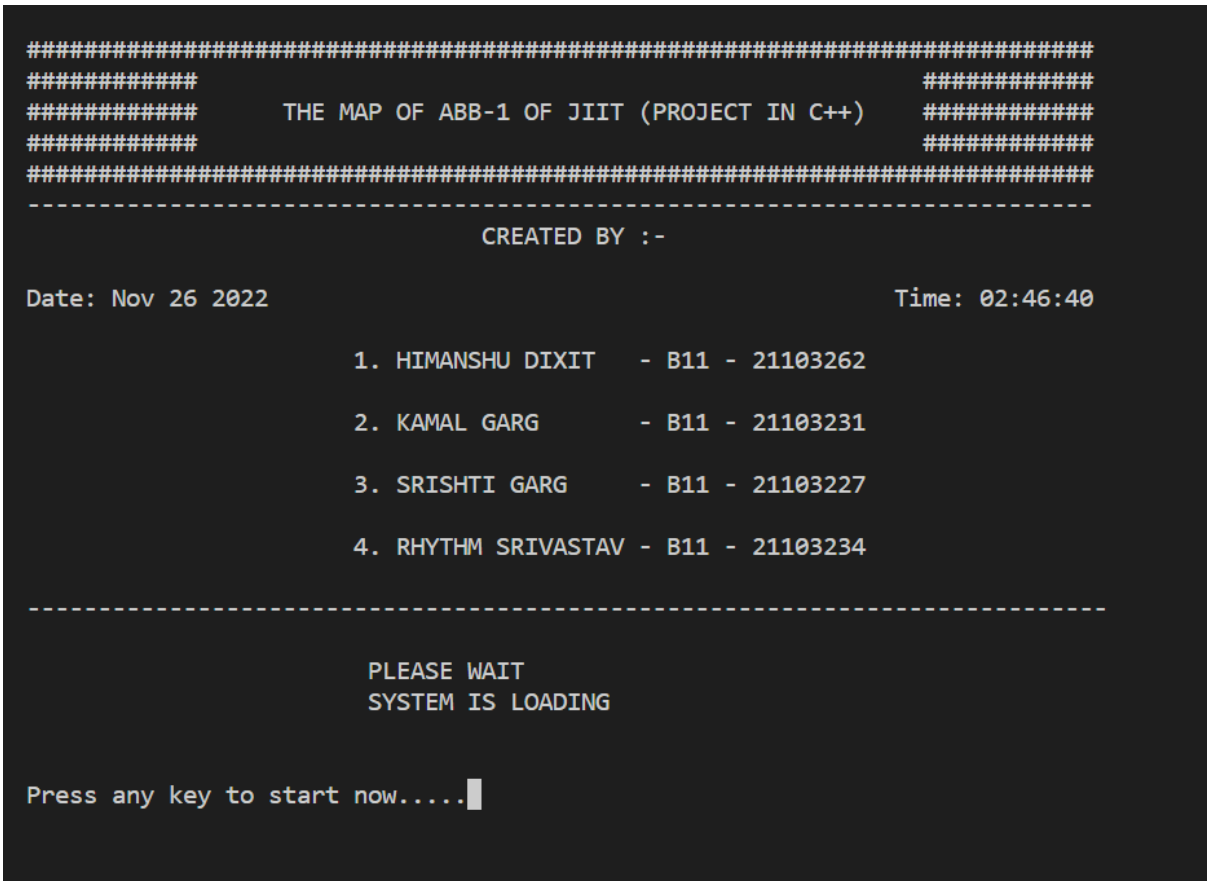
```
    help(38, 37, 4);
    help(36, 33, 3);
    help(33, 36, 3);
    help(36, 35, 3);
    help(35, 36, 3);
    help(35, 34, 3);
    help(34, 35, 3);
    help(33, 34, 3);
    help(34, 33, 3);
    help(34, 28, 5);
    help(28, 34, 5);
    help(36, 38, 3);
    help(38, 36, 3);
    help(40, 41, 1);
    help(41, 40, 1);
    help(40, 71, 1);
    help(71, 40, 1);
    help(41, 42, 2);
    help(42, 41, 2);
    help(42, 43, 2);
    help(43, 42, 2);
    help(43, 44, 2);
    help(44, 43, 2);
    help(44, 45, 1);
    help(45, 44, 1);
    help(44, 62, 4);
    help(62, 44, 4);
    help(45, 46, 2);
    help(46, 45, 2);
    help(46, 47, 2);
    help(47, 46, 2);
    help(47, 48, 2);
    help(48, 47, 2);
    help(48, 49, 1);
    help(49, 48, 1);
    help(49, 59, 4);
    help(59, 49, 4);
    help(59, 60, 1);
    help(60, 59, 1);
    help(59, 61, 6);
    help(61, 59, 6);
    help(49, 50, 2);
    help(50, 49, 2);
    help(50, 51, 2);
    help(51, 50, 2);
    help(51, 52, 2);
    help(52, 51, 2);
    help(52, 53, 1);
    help(53, 52, 1);
    help(53, 54, 4);
    help(54, 53, 4);
    help(54, 55, 2);
    help(55, 54, 2);
    help(55, 56, 2);
    help(56, 55, 2);
    help(56, 57, 1);
    help(57, 56, 1);
    help(56, 58, 3);
    help(58, 56, 3);
    help(62, 63, 1);
    help(63, 62, 1);
    help(63, 64, 2);
```

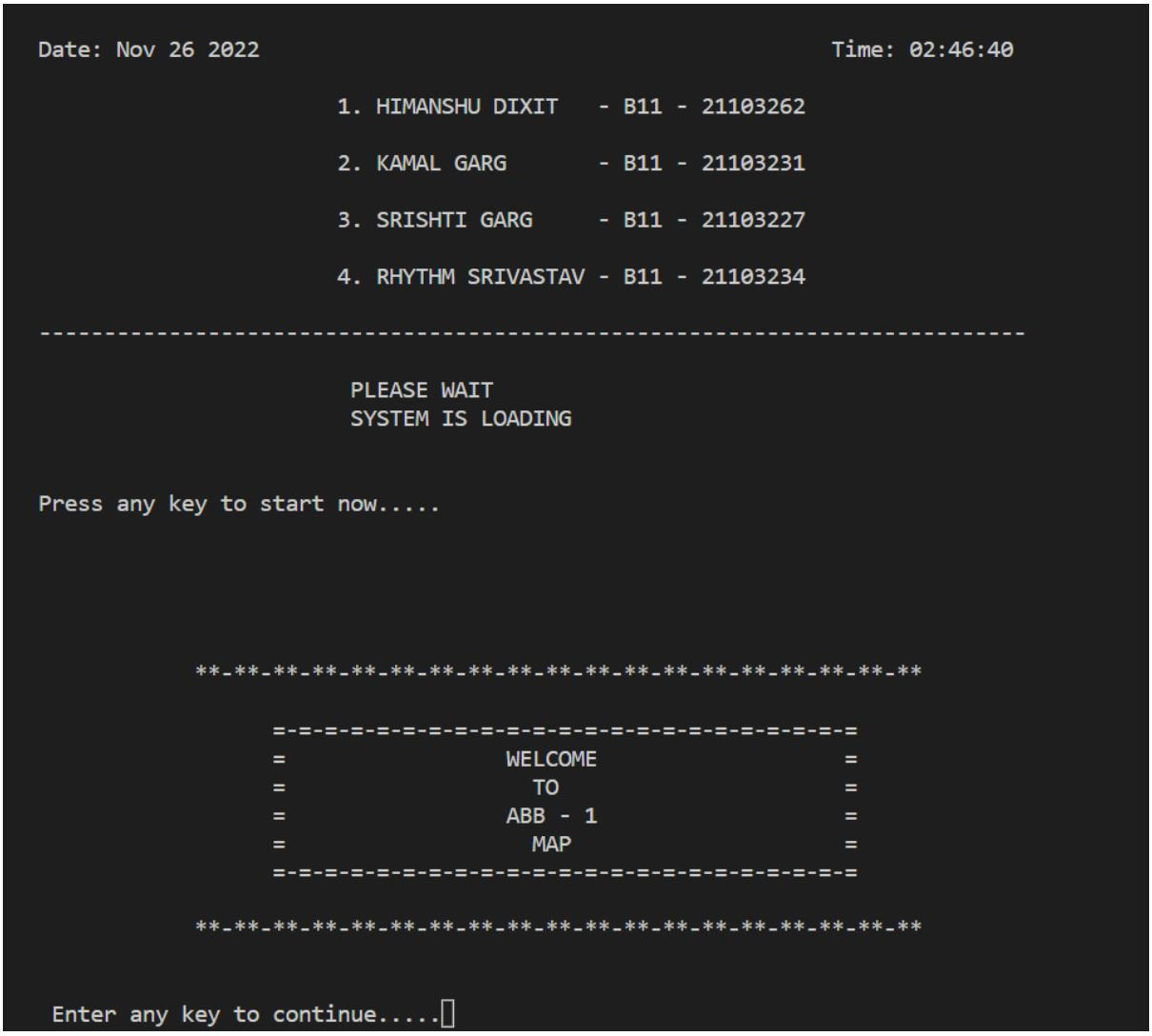
```
    help(64, 63, 2);
    help(64, 65, 3);
    help(65, 64, 3);
    help(65, 66, 3);
    help(66, 65, 3);
    help(66, 67, 2);
    help(67, 66, 2);
    help(66, 68, 3);
    help(68, 66, 3);
    help(67, 68, 3);
    help(68, 67, 3);
    help(69, 68, 2);
    help(68, 69, 2);
    help(70, 69, 1);
    help(69, 70, 1);
    help(70, 71, 2);
    help(71, 70, 2);
    help(72, 73, 1);
    help(73, 72, 1);
    help(73, 74, 3);
    help(74, 73, 3);
    help(74, 75, 2);
    help(75, 74, 2);
    help(72, 90, 1);
    help(90, 72, 1);
    help(75, 76, 2);
    help(76, 75, 2);
    help(76, 77, 3);
    help(77, 76, 3);
    help(77, 78, 2);
    help(78, 77, 2);
    help(78, 79, 1);
    help(79, 78, 1);
    help(79, 80, 3);
    help(80, 79, 3);
    help(80, 87, 1);
    help(81, 82, 2);
    help(82, 83, 1);
    help(83, 107, 2);
    help(83, 105, 4);
    help(105, 106, 3);
    help(105, 104, 3);
    help(103, 102, 3);
    help(102, 101, 4);
    help(102, 86, 4);
    help(86, 88, 1);
    help(86, 85, 1);
    help(88, 87, 1);
    help(87, 78, 3);
    help(85, 84, 2);
    help(84, 78, 1);
    help(75, 89, 4);
    help(89, 96, 1);
    help(96, 97, 2);
    help(97, 98, 1);
    help(98, 99, 3);
    help(98, 100, 3);
    help(90, 91, 2);
    help(91, 92, 1);
    help(92, 93, 2);
    help(93, 94, 2);
    help(94, 95, 3);
```

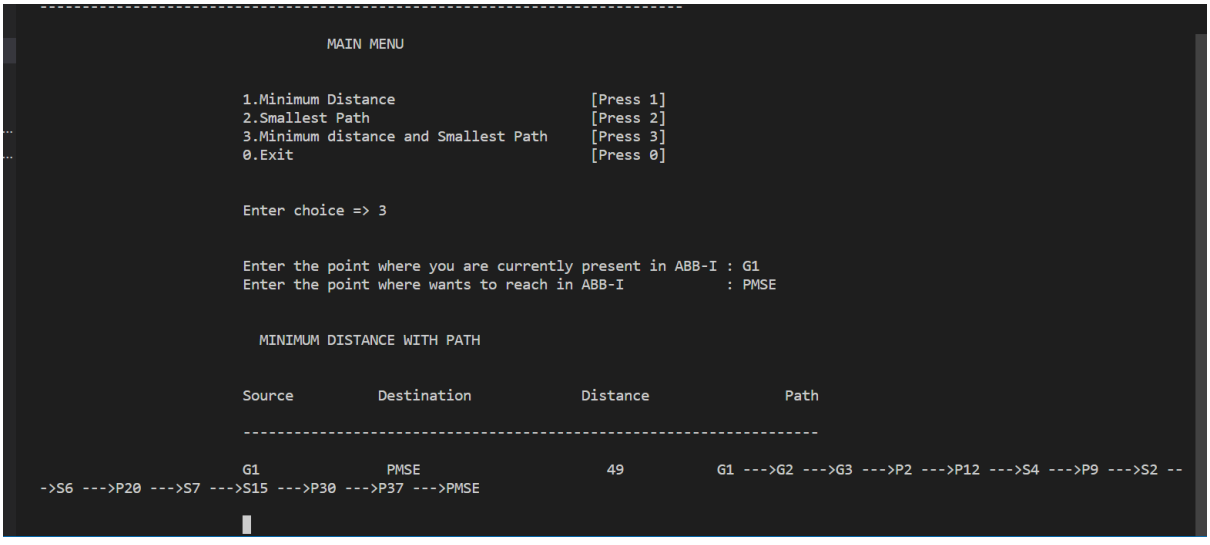
```
    help(1, 108, 4);
    help(108, 37, 2);
    help(109, 5, 2);
    help(110, 10, 2);
    help(111, 14, 2);
    help(69, 37, 8);
    help(63, 37, 8);
    help(57, 107, 11);
    help(96, 63, 8);
    help(97, 63, 8);
    help(93, 69, 8);
    help(55, 22, 8);
    help(9, 60, 7);
    help(60, 87, 7);
    help(60, 87, 7);

    do
    {
        headMessage("CREATED BY :-");
        printMessageCenter2("MAIN MENU");
        cout << "\n\n\t\t\t1.Minimum Distance                    [Press 1]";
        cout << "\n\t\t\t2.Smallest Path                       [Press 2]";
        cout << "\n\t\t\t3.Minimum distance and Smallest Path  [Press 3]";
        cout << "\n\t\t\t0.Exit                                [Press 0]";
        cout << "\n\n\n\t\t\tEnter choice => ";
        cin >> choice;

        if (choice >= 1 && choice <= 3)
        {

            cout << "\n\n\t\t\tEnter the point where you are currently present in ABB-I : ";
            cin >> s;
            cout << "\t\t\tEnter the point where wants to reach in ABB-I           : ";
            cin >> d;

            // this function is searching the index of points

            i = j = 0;
            for (int i = 0; i < 112; i++)
            {
                if (names[i] == s)
                    source = i;
                if (names[i] == d)
                    destination = i;
            }
        }

        switch (choice)
        {
        case 1:
            a = 0;
            cout << "\n\n";
            printMessageCenter2("MINIMUM DISTANCE");
            cout << dijkstra(arr, source, destination, choice) << endl;
            cout << "\n\n\t\t\t";
            getch();

            break;
        case 2:
            a = 0;
            cout << "\n\n";
            printMessageCenter2("SMALLEST PATH");
```

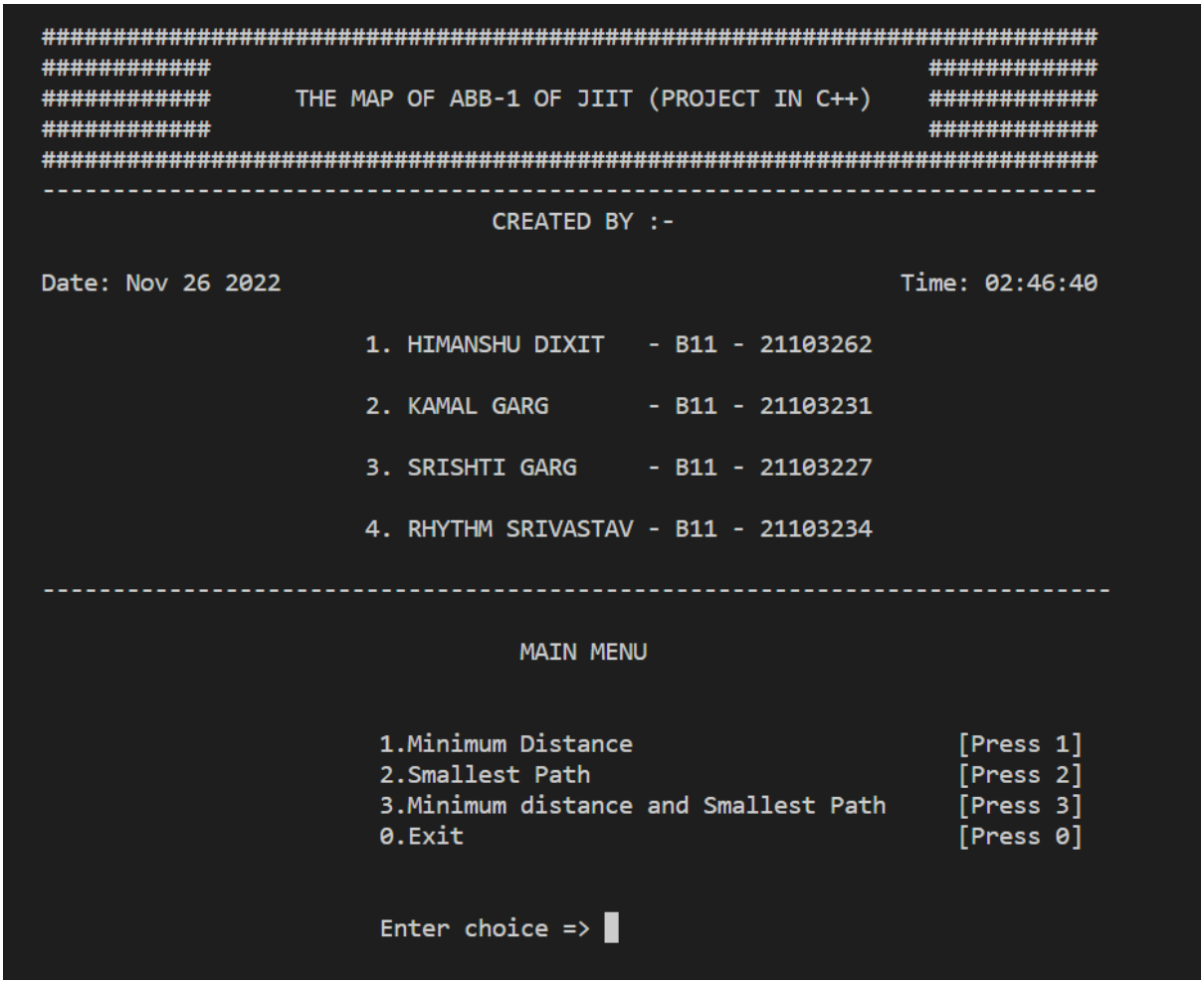
```
            dijkstra(arr, source, destination, choice);
            printPath(destination, parents);
            cout << "\n\n\t\t\t";
            getch();
            break;
        case 3:
            a = 0;
            cout << "\n\n";
            printMessageCenter2("MINIMUM DISTANCE WITH PATH");
            cout << dijkstra(arr, source, destination, choice) << "\t\t";
            printPath(destination, parents);
            cout << "\n\n\t\t\t";
            getch();
            break;
        case 0:
            printf("\n\n\n\t\t\t\tThank you!!!\n\n\n\n\n");
            exit(1);
            break;
        default:
            printf("\n\n\n\t\t\tINVALID INPUT!!! Try again...");
        }
        // Switch Ended
    } while (choice != 0);
    // Loop Ended
}

//-----------------------------------------------------------------------------------------------
-----------
// The Program starts from here
//-----------------------------------------------------------------------------------------------
-----------

int main()
{
    Welcome_Msg();
    Menu();
    return 0;
}
```

OUTPUTS SCREENSHOTS:




Date: Nov 26 2022
Time: 02:46:40
1. HIMANSHU DIXIT - B11 - 21103262
2. KAMAL GARG - B11 - 21103231
3. SRISHTI GARG - B11 - 21103227
4. RHYTHM SRIVASTAV - B11 - 21103234
PLEASE WAIT
SYSTEM IS LOADING
Press any key to start now.....
WELCOME
TO
ABB - 1
MAP
Enter any key to continue.....


MAIN MENU
1.Minimum Distance [Press 1]
2.Smallest Path [Press 2]
3.Minimum distance and Smallest Path [Press 3]
0.Exit [Press 0]
Enter choice => 3
Enter the point where you are currently present in ABB-I : G1
Enter the point where wants to reach in ABB-I : PMSE
MINIMUM DISTANCE WITH PATH
Source Destination Distance Path
G1 PMSE 49 G1 --->G2 --->G3 --->P2 --->P12 --->S4 --->P9 --->S2 --
->S6 --->P20 --->S7 --->S15 --->P30 --->P37 --->PMSE


THE MAP OF ABB-1 OF JIIT (PROJECT IN C++)
CREATED BY :-
Date: Nov 26 2022
Time: 02:46:40
1. HIMANSHU DIXIT - B11 - 21103262
2. KAMAL GARG - B11 - 21103231
3. SRISHTI GARG - B11 - 21103227
4. RHYTHM SRIVASTAV - B11 - 21103234
MAIN MENU
1.Minimum Distance [Press 1]
2.Smallest Path [Press 2]
3.Minimum distance and Smallest Path [Press 3]
0.Exit [Press 0]
Enter choice =>