VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590010

MINI PROJECT REPORT

ON

"FLIGHT MINIMUM COST"

Submitted in partial fulfillment for the requirements for the seventh semester curriculum

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

For the Academic year 2021-2022

Submitted by:

BHARATH GOWDA R	**1MV20IS015**
KUPPA SHASHANK	**1MV20IS026**
NADIMPALLI KUSHAL	**1MV20IS032**
PIYUSH NAHAR	**1MV20IS035**

Project carried out at:

Sir M. Visvesvaraya Institute of Technology

Bengaluru-562157

Under the guidance of:

DR.C.H. VANIPRIYA

Assistant Professor, Department of ISE

Sir M. Visvesvaraya Institute of Technology, Bengaluru

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING SIR. M VISVESVARAYA INSTITUTE OF TECHNOLOGY HUNASAMARANAHALLI, BENGALURU-562157

DECLARATION

We hereby declare that the entire mini project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bengaluru

Date:

Signature of Students:

BHARATH GOWDA R
(1MV20IS015)

KUPPA SHASHANK

(1MV20IS026)

NADIMPALLI KUSHAL

(1MV20IS032)

PIYUSH NAHAR
(1MV20IS035)

ABSTRACT:

The goal of this mini-project is to find the minimum cost of each destination and the air route to be taken for it from any given destination.

In this program we have given three (example)sample air routes for three different air routes and the cost of their tickets. These information is stored in the program in the form of adjacency matrix. The program then asks for the source and destination of the customer and then it runs dijkstra's algorithm to find the minimum cost and it also keeps track while traversing through the graph, in order to know the air route.
Then based on the customer's choice it displays the minimum cost and the route that should be taken.
If there are no routes available for any particular destination in a selected airline then customers can check it with other airlines.
Customers can compare the cost availability across airlines and also campare with various destinations too if they r finding it difficult to select a vacation destination , as these information can help them making their decisions easier.

TABLE OF CONTENTS

Chapter 1 - INTRODUCTION

Introduction to C and Data Structures

C is a general-purpose, procedural computer programming language supporting structured programming, lexical variable scope, and recursion, with a static type system. By design, C provides constructs that map efficiently to typical machine instructions. It has found lasting use in applications previously coded in assembly language. Such applications include operating systems and various application software for computer architectures that range from supercomputers to PLCs and embedded systems.

A successor to the programming language *B*, C was originally developed at Bell Labs by Dennis Ritchie between 1972 and 1973 to construct utilities running on Unix. It was applied to re-implementing the kernel of the Unix operating system. During the 1980s, C gradually gained popularity. It has become one of the most widely used programming languages, with C compilers from various vendors available for the majority of existing computer architectures and operating systems. C has been standardized by the ANSI since 1989 (ANSI C) and by the International Organization for Standardization (ISO).

C is an imperative procedural language. It was designed to be compiled to provide low-level access to memory and language constructs that map efficiently to machine instructions, all with minimal runtime support. Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant C program written with portability in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

INTRODUCTION TO DATA STRUCTURES

Data Structure can be defined as the group of data elements which provides an efficient way of storing and organising data in the computer so that it can be used efficiently. Some examples of Data Structures are arrays, Linked List, Stack, Queue, etc. Data Structures are widely used in almost every aspect of Computer Science i.e. Operating System, Compiler Design, Artificial intelligence, Graphics and many more.

Data Structures are the main part of many computer science algorithms as they enable the programmers to handle the data in an efficient way. It plays a vital role in enhancing the performance of a software or a program as the main function of the software is to store and retrieve the user's data as fast as possible.

Chapter 2 – SYSTEM REQUIREMENTS

2.2 Hardware requirement

C programming can be basically run on any workstation which has UNIX system, MS-DOS
system, Windows system, UNIX like Solaris, IRIX, GNU/Linux, BSD, Mac OS, Fedora.
All u need is as a minimum, you should have at least amount space for data and a processing RAM of good quality requirement is enough run a c programming in a system. A keyboard for a system and mouse is a basic necessity for handling a system.

2.3 Software requirement

There is a large amount of software on which we can run a C program. To construct a C program all we need is a **text editor** and a **C compiler**. All we need the files we create with our editor are called the source files and they contain the program source codes. The source files for C programs are typically named with the extension "**.c**".

The source code written in source file is the human readable source for your program. It needs to be "compiled", into machine language so that your CPU can actually execute the program as per the instructions given. The compiler compiles the source codes into final executable programs. The most frequently used and free available compiler is the turbo C++, otherwise you can have compilers either from HP or Solaris if you have the respective operating systems.

Chapter:3 SYSTEM ANALYSIS

It is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. **Systems analysis** is "the process of studying a procedure or business in order to identify its goals and purposes and create systems and procedures that will achieve them in an efficient way". Another view sees system analysis as a problem-solving technique that breaks down a system into its component pieces for the purpose of the studying how well those component parts work and interact to accomplish their purpose.

The field of system analysis relates closely to requirements analysis or to operations research. It is also "an explicit formal inquiry carried out to help a decision maker identify a better course of action and make a better decision than they might otherwise have made."

The terms analysis and synthesis stem from Greek, meaning "to take apart" and "to put together," respectively. These terms are used in many scientific disciplines, from mathematics and logic to economics and psychology, to denote similar investigative procedures. Analysis is defined as "the procedure by which we break down an intellectual or substantial whole into parts," while synthesis means "the procedure by which we combine separate elements or components in order to form a coherent whole." System analysis researchers apply methodology to the systems involved, forming an overall picture.

System analysis is used in every field where something is developed. Analysis can also be a series of components that perform organic functions together, such as system engineering. System engineering is an interdisciplinary of engineering that focuses on how complex engineering projects should be designed and manage.

Chapter4:
SYSTEM DESIGN

4.1 Activities of DESIGN Process/System Architecture

By the end of this project, you will create a fully functioning “flight minimum cost” system, this will be achieved through applying and practicing many concepts of programming which programmers use all the time through their programming careers such as advanced if statements, advancedarithmetic operations, loops, Arrays, and 2D arrays, Dynamic memory, Data Structures. By applying these concepts, you will also be able to create different types of programs that users can interact with.
These programming concepts can also be applied using other Programming Languages such as Java and Python, not just C++.

1. Introduction.
2. Applications on If-statements
3. Increment & Decrement Operations
4. While loops & Do-While loops
5. For loops & Nested loops
6. Arrays and 2D Arrays
7. Dynamic memory allocations
8. Graphs using adjacency matrix
9. Application of dijkstra’s algorithm

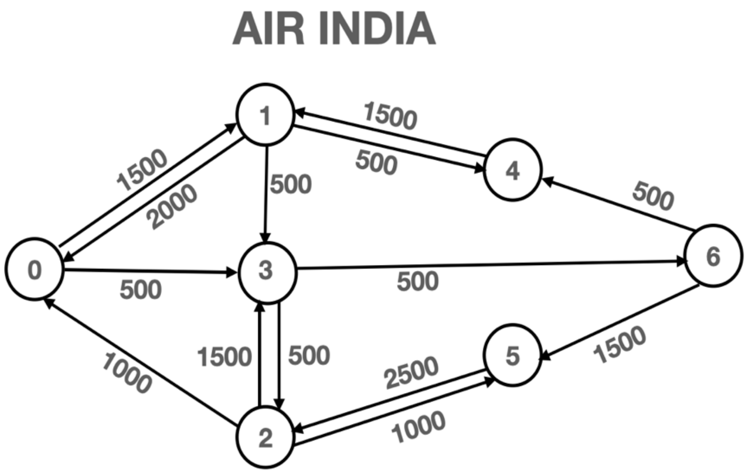
Chapter: 5

PROJECT MODULE DESCIPTION/

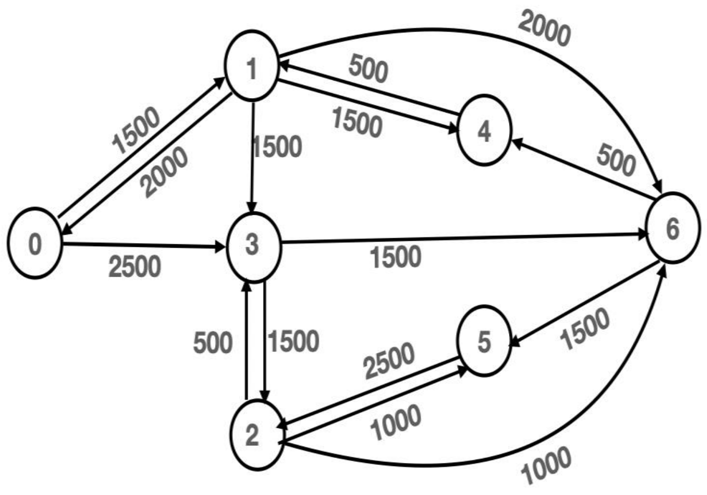
IMPLEMENTATION

```
#include<stdio.h>
#include<stdlib.h>

#define True 1
#define False 0
#define infinity 999999
#define n 7


int *graph1,*graph2,*graph3;

int track[n];

void dijkstra(int graph[], int s,int cost[],int visited[]){
    int b=0;
    for(int i=0;i<n;i++){
        if(graph[(s*n)+i]+cost[s]<cost[i]){
            cost[i]=graph[(s*n)+i]+cost[s];
            track[i]=s;
        }
        if(cost[i]<infinity && !visited[i]){
            b=i;
        }
    }
    if(!visited[b]){
        visited[b]=True;
        dijkstra(graph,b,cost,visited);
        return;
    }
    return;
}

void path(int i, int s){
    if(i==s)
        return;
    printf("%d<-",i);
    path(track[i],s);
}
```

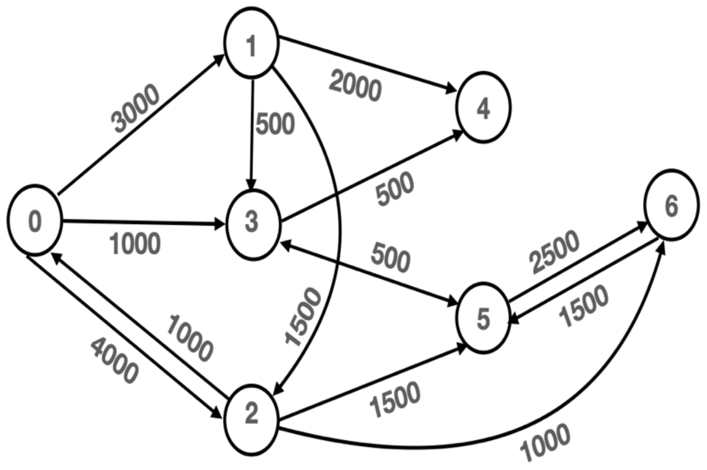
```
void display(int cost[], int s){
    for(int i=0;i<n;i++){
        if(cost[i]!=infinity && i!=s){
            printf("\nThe minimum cost for %d: %d\n", i, cost[i]);
            path(i,s);
            printf("%d\n",s);
        }
        else if(cost[i]==infinity) {
            printf("\nNo flight paths are available for destination %d in this airline. Please select a
different airline.\n",i);
        }
    }
}


void display2(int cost[],int s, int d){
    if(cost[d]!=infinity && d!=s){
            printf("\nThe minimum cost for %d: %d\n", d, cost[d]);
            path(d,s);
            printf("%d\n",s);
        }
    else if(cost[d]==infinity) {
        printf("\nNo flight paths are available for destination %d in this airline. Please select a
different airline.\n",d);
        }
    else
        printf("\nInvalid Input\n");
}



int main(){
    while(1){
    int ch,dest;
    graph1 = malloc(n*n*sizeof(int));
    graph2 = malloc(n*n*sizeof(int));
    graph3 = malloc(n*n*sizeof(int));
    int i,j,cost,s;
    int visited1[n],visited2[n],visited3[n];
    int cost1[n],cost2[n],cost3[n];

    printf("\nEnter the airline:\n1.Air India\n2.Indigo\n3.Jet Airways\n4.Exit\n");
    scanf("%d", &ch);
    switch(ch){
        case 1:
            for(int h=0;h<n;h++){
            visited1[h]=False;
            cost1[h]=infinity;
            }

            for(int j=0;j<n*n;j++){
                graph1[j]=infinity;
            }

            graph1[(0*n)+1]=1500;
            graph1[(0*n)+3]=500;
            graph1[(1*n)+0]=2000;
            graph1[(1*n)+3]=500;
            graph1[(1*n)+4]=500;
            graph1[(1*n)+6]=2000;
            graph1[(2*n)+0]=1000;
            graph1[(2*n)+3]=1500;
            graph1[(2*n)+5]=1000;
            graph1[(3*n)+2]=500;
            graph1[(3*n)+6]=500;
            graph1[(4*n)+1]=1500;
            graph1[(5*n)+2]=2500;
            graph1[(6*n)+4]=500;
            graph1[(6*n)+5]=1500;

            printf("\nEnter the
source:\n0.Bengaluru\n1.Chennai\n2.Mumbai\n3.Delhi\n4.Kolkata\n5.Hyderabad\n6.Ahmeda
bad\n");
            scanf("%d", &s);
            graph1[(s*n)+s]=0;
            visited1[s]=True;
            cost1[s]=0;
            dijkstra(graph1,s,cost1,visited1);
            printf("\nChoose the
Destination:\n0.Bengaluru\n1.Chennai\n2.Mumbai\n3.Delhi\n4.Kolkata\n5.Hyderabad\n6.Ah
medabad\n7.All Destinations\n");
            scanf("%d", &dest);
            if(dest==7)
                display(cost1, s);
            else
                display2(cost1, s, dest);
            break;

        case 2:
            for(int h=0;h<n;h++){
                visited2[h]=False;
                cost2[h]=infinity;
            }
            for(int j=0;j<n*n;j++){
                graph2[j]=infinity;
            }
                        graph2[(0*n)+1]=1500;
            graph2[(0*n)+3]=2500;
            graph2[(1*n)+0]=2000;
            graph2[(1*n)+3]=1500;
            graph2[(1*n)+4]=1500;
            graph2[(1*n)+6]=2000;
            graph2[(2*n)+6]=1000;
            graph2[(2*n)+3]=500;
            graph2[(2*n)+5]=1000;
            graph2[(3*n)+2]=1500;
            graph2[(3*n)+6]=1500;
            graph2[(4*n)+1]=500;
            graph2[(5*n)+2]=2500;
            graph2[(6*n)+4]=500;
            graph2[(6*n)+5]=1500;

            printf("\nEnter the 
source:\n0.Bengaluru\n1.Chennai\n2.Mumbai\n3.Delhi\n4.Kolkata\n5.Hyderabad\n6.Ahmeda
bad\n");
            scanf("%d", &s);
            graph2[(s*n)+s]=0;
            visited2[s]=True;
            cost2[s]=0;
            dijkstra(graph2,s,cost2,visited2);
            printf("\nChoose the 
Destination:\n0.Bengaluru\n1.Chennai\n2.Mumbai\n3.Delhi\n4.Kolkata\n5.Hyderabad\n6.Ah
medabad\n7.All Destinations\n");
            scanf("%d", &dest);
            if(dest==7)
                display(cost2, s);
            else
                display2(cost2, s, dest);
            break;

        case 3:
            for(int h=0;h<n;h++){
                visited3[h]=False;
                cost3[h]=infinity;
            }
            for(int j=0;j<n*n;j++){
                graph3[j]=infinity;
            }
                        graph3[(0*n)+1]=3000;
            graph3[(0*n)+2]=4000;
            graph3[(0*n)+3]=1000;
            graph3[(1*n)+2]=1500;
            graph3[(1*n)+3]=500;
            graph3[(1*n)+4]=2000;
            graph3[(2*n)+0]=1000;
            graph3[(2*n)+5]=1500;
            graph3[(2*n)+6]=1000;
            graph3[(3*n)+4]=500;
            graph3[(3*n)+5]=500;
            graph3[(4*n)+5]=1500;
            graph3[(5*n)+6]=2500;
            graph3[(5*n)+3]=500;
            graph3[(6*n)+5]=1500;
            printf("\nEnter the 
source:\n0.Bengaluru\n1.Chennai\n2.Mumbai\n3.Delhi\n4.Kolkata\n5.Hyderabad\n6.Ahmeda
bad\n");
            scanf("%d", &s);
            graph3[(s*n)+s]=0;
            visited3[s]=True;
            cost3[s]=0;
            dijkstra(graph3,s,cost3,visited3);
            printf("\nChoose the
Destination:\n0.Bengaluru\n1.Chennai\n2.Mumbai\n3.Delhi\n4.Kolkata\n5.Hyderabad\n6.Ah
medabad\n7.All Destinations\n");
            scanf("%d", &dest);
            if(dest==7)
                display(cost3, s);
            else
                display2(cost3, s, dest);
            break;

        default:
            exit(0);

        }

    }

    return 0;
}
```



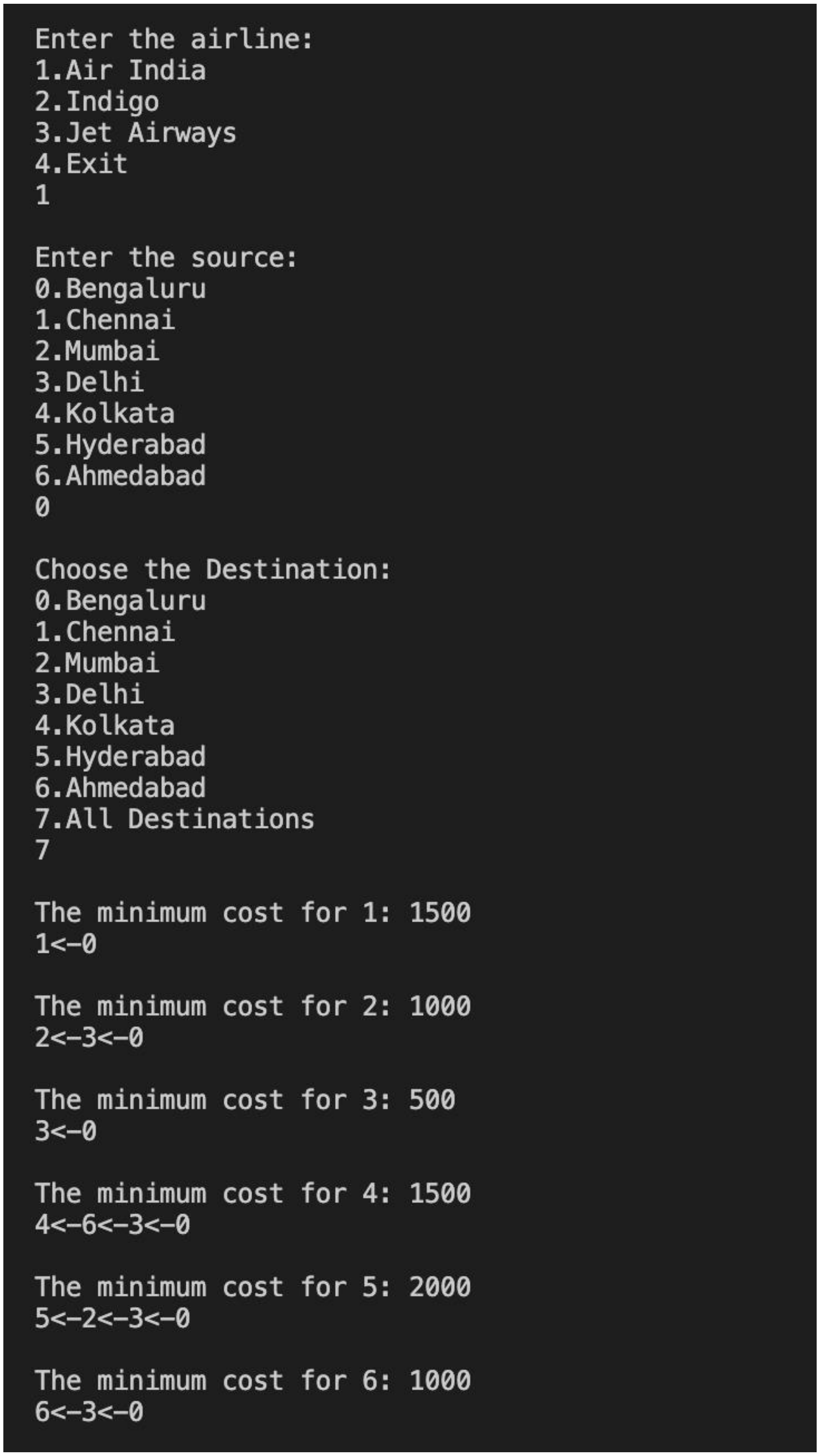
INDIGO



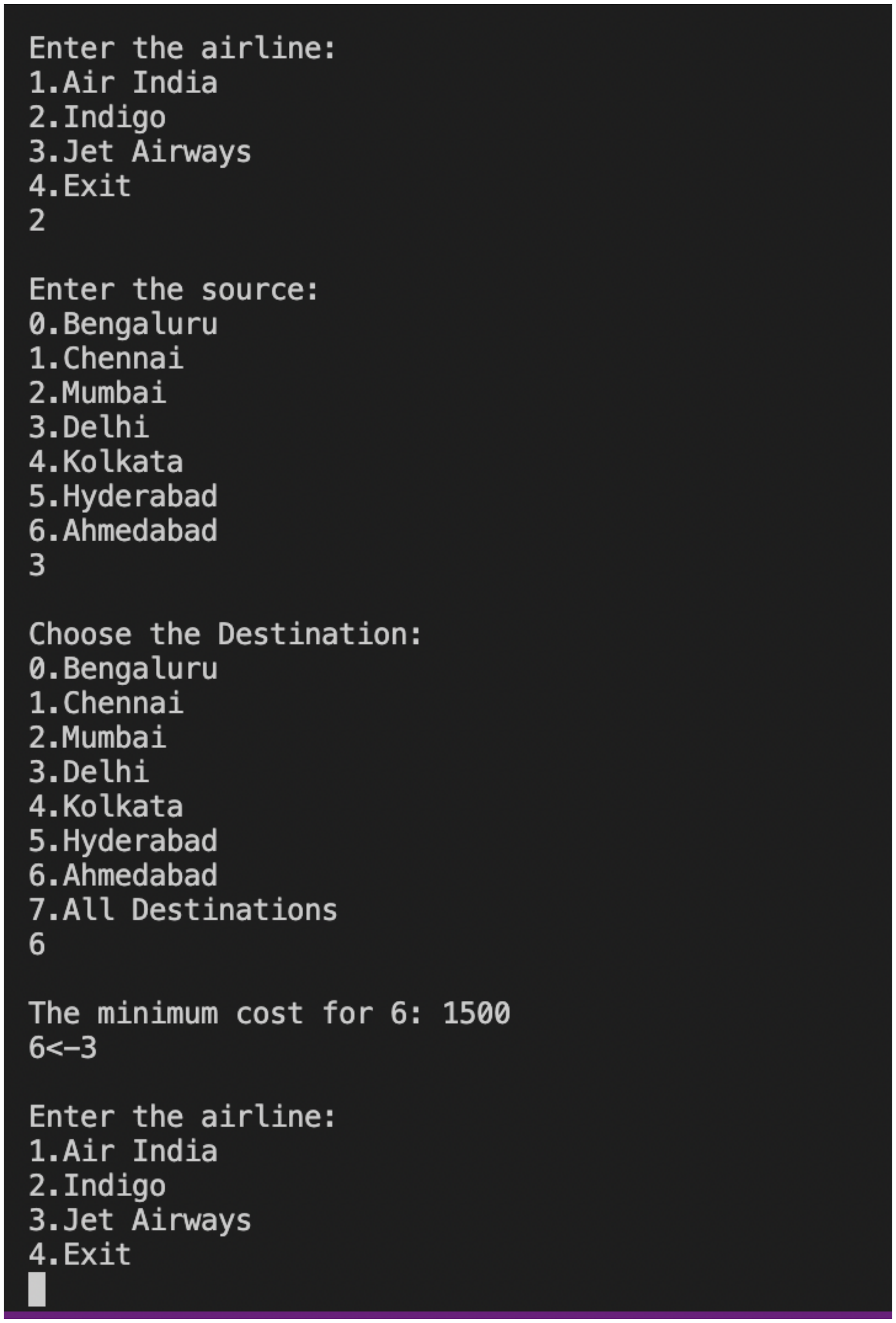
JET AIRWAYS



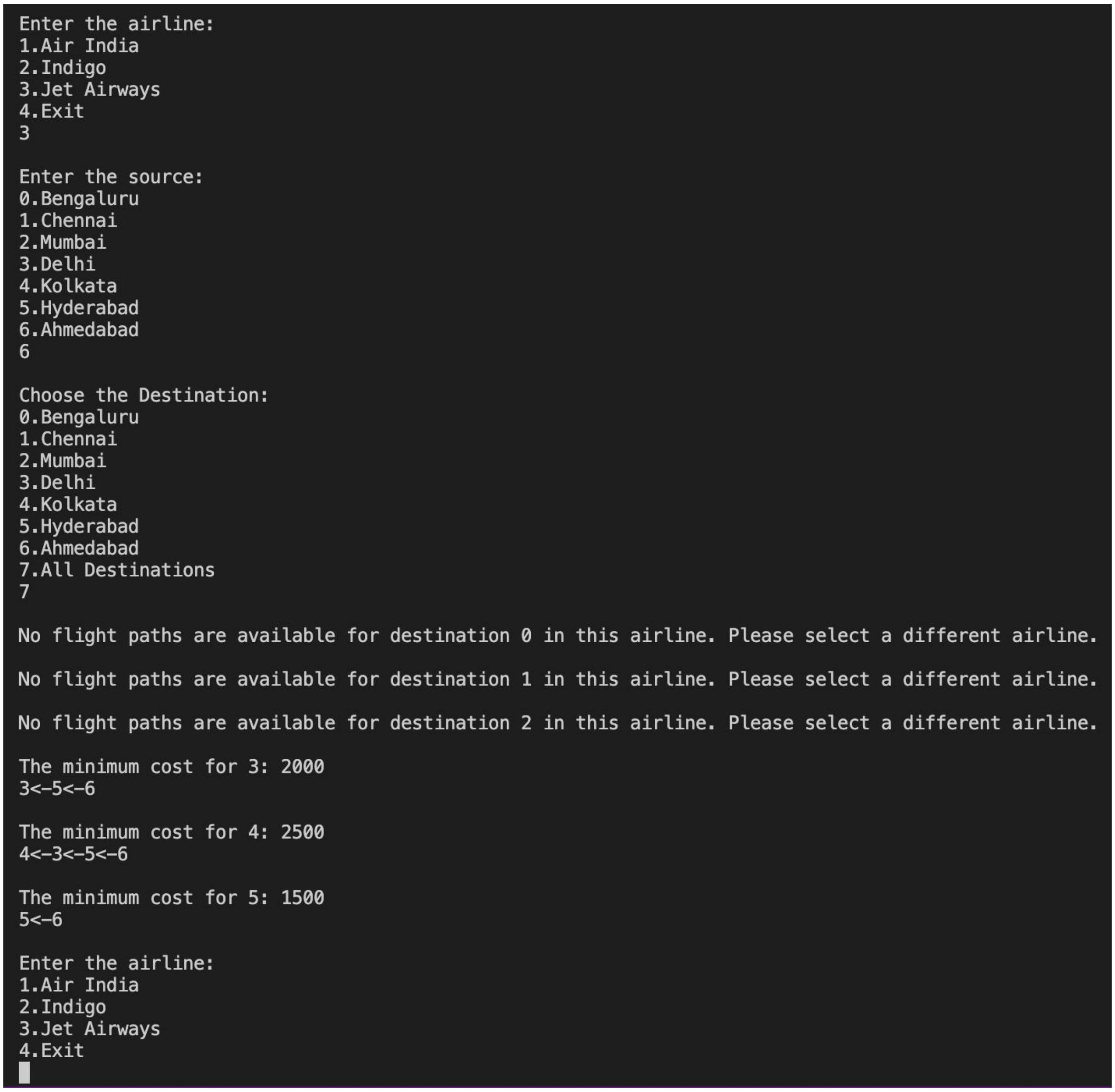
OUTPUT:



```
Enter the airline:
1.Air India
2.Indigo
3.Jet Airways
4.Exit
1

Enter the source:
0.Bengaluru
1.Chennai
2.Mumbai
3.Delhi
4.Kolkata
5.Hyderabad
6.Ahmedabad
0

Choose the Destination:
0.Bengaluru
1.Chennai
2.Mumbai
3.Delhi
4.Kolkata
5.Hyderabad
6.Ahmedabad
7.All Destinations
7

The minimum cost for 1: 1500
1<-0

The minimum cost for 2: 1000
2<-3<-0

The minimum cost for 3: 500
3<-0

The minimum cost for 4: 1500
4<-6<-3<-0

The minimum cost for 5: 2000
5<-2<-3<-0

The minimum cost for 6: 1000
6<-3<-0
```



```
Enter the airline:
1.Air India
2.Indigo
3.Jet Airways
4.Exit
2

Enter the source:
0.Bengaluru
1.Chennai
2.Mumbai
3.Delhi
4.Kolkata
5.Hyderabad
6.Ahmedabad
3

Choose the Destination:
0.Bengaluru
1.Chennai
2.Mumbai
3.Delhi
4.Kolkata
5.Hyderabad
6.Ahmedabad
7.All Destinations
6

The minimum cost for 6: 1500
6<-3

Enter the airline:
1.Air India
2.Indigo
3.Jet Airways
4.Exit
```



```
Enter the airline:
1.Air India
2.Indigo
3.Jet Airways
4.Exit
3

Enter the source:
0.Bengaluru
1.Chennai
2.Mumbai
3.Delhi
4.Kolkata
5.Hyderabad
6.Ahmedabad
6

Choose the Destination:
0.Bengaluru
1.Chennai
2.Mumbai
3.Delhi
4.Kolkata
5.Hyderabad
6.Ahmedabad
7.All Destinations
7

No flight paths are available for destination 0 in this airline. Please select a different airline.

No flight paths are available for destination 1 in this airline. Please select a different airline.

No flight paths are available for destination 2 in this airline. Please select a different airline.

The minimum cost for 3: 2000
3<-5<-6

The minimum cost for 4: 2500
4<-3<-5<-6

The minimum cost for 5: 1500
5<-6

Enter the airline:
1.Air India
2.Indigo
3.Jet Airways
4.Exit
```

Chapter:6

TESTING

Testing means verifying correct behavior. Testing can be done at all stages of module development: requirements analysis, interface design, algorithm design, implementation, and integration with other modules. The attention will be directed at implementation testing. Implementation testing is not restricted to execution testing. An implementation can also be tested using correctness proofs, code tracing, and peer reviews, as described below.

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure.

Chapter 7:

BIBLIOGRAPHY

1. Ellis Horowitz and Sartaj Sahini, Fundamentals of Data Structures in C
2. https://youtu.be/smHnz2RHJBY
3. Geeksforgeeks website.