Introduction to The Language Used – C++

ABOUT THE LANGUAGE

C++ is a general-purpose object-oriented programming (OOP) language, developed by **Bjarne Stroustrup**, and is an extension of the C language. It is therefore possible to code C++ in a "C style" or "object-oriented style." In certain scenarios, it can be coded in either way and is thus an effective example of a hybrid language.

C++ is considered to be an intermediate-level language, as it encapsulates both high- and low-level language features. Initially, the language was called "C with classes" as it had all the properties of the C language with an additional concept of "classes." However, it was renamed C++ in 1983.

C++ is a highly portabl*e* language and is often the language of choice for multi-device, multi-platform app development.

- It is an object-oriented programming language and includes classes, inheritance, polymorphism, data abstraction and encapsulation.
- It has a rich function library.
- It allows exception handling, and function overloading which are not possible in C.

OBJECT ORIENTED PROGRAMMING

The prime purpose of C++ programming was to add object orientation to the C programming language, which is in itself one of the most powerful programming languages.

The core of the pure object-oriented programming is to create an object, in code, that has certain properties and methods. While designing C++ modules, we try to see whole world in the form of objects. For example a car is an object which has certain properties such as color, number of doors, and the like. It also has certain methods such as accelerate, brake, and so on.

Advantages of OOPS Concepts-

1. OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.
2. OOPs provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.
3. OOPs provide ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

There are a few principle concepts that form the foundation of object-oriented programming

Data Abstraction

Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

For example, a database system hides certain details of how data is stored and created and maintained. Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.

Data Encapsulation

Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.

Object:

This is the basic unit of object oriented programming. That is both data and function that operate on data are bundled as a unit called as object.

Class

When you define a class, you define a blueprint for an object. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

Inheritance

One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

This is a very important concept of object-oriented programming since this feature helps to reduce the code size.

Polymorphism

The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism. Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.

Uses/Applications of C++ Language

- Operating Systems
- Browsers
- Libraries
- Graphics
- Banking Applications
- Databases
- Embedded Systems

Disadvantages of C++

- It has no security.
- Complex in a very large high-level program.
- Used for platform specific application commonly.
- When **C++** used for web applications complex and difficult to debug.

About the Project

Snakes and Ladders is an ancient Indian game regarded today as a worldwide classic. The game is a simple race contest based on sheer luck. It is played between two or more players on a game board having numbered, gridded squares.

A number of "ladders" and "snakes" are pictured on the board, each connecting two specific board squares. The object of the game is to navigate one's game piece(token), according to die rolls, from the start (bottom square) to the finish (top square), helped or hindered by ladders and snakes respectively. When you go up a ladder, you progress quickly nearing the finish. When you go down a snake, you go backwards near the start.

In this project, the ancient game of pawns and board has been implemented virtually, where the computer generates the number on the dice and the tokens on the screen are moved accordingly.

History:

Also known as Chutes and Ladders, Snakes and Ladders was a game that was designed to teach morality. The game was initially devised to teach principles of virtue, represented by the ladders, and evil, represented by snakes. The goal of the game was to reach spiritual nirvana. Squares that were titled faith, generosity, etc. bumped the players up the ladders whereas squares such as disobedience, debt, drunkenness, etc. ensured the player tumbled down.

How to play:

Each player has their token @/# at the starting position or null position. Take it in turns to roll the dice. your token is moved forward the number of spaces shown on the dice. If your token lands at the bottom of a ladder, you can move up to the top of the ladder. If your token lands on the head of a snake, you must slide down to the bottom of the snake. The first player to get to the space that reads '100' is the winner. Have fun!

Number of players: 2

Advantages for children:

It's perhaps no surprise that Snakes and Ladders develops children counting abilities. The interesting thing here is that Snakes and Ladders not only develops counting skills but also basic addition skills. For instance, if a child is on square 9 and receive a dice roll of 5 he/she will move forward 5 spaces to 14. The realization will dawn on the child, or can be prompted by an adult that 9 + 5 = 14. They will consistently be rolling fours, five and sixes, and counting the numbers on the board. Children also learn the basic sequence and pattern of numbers.

Teaching life lessons to children:

The game helps children develop social language skills in that bad things can happen suddenly and without warning. For instance, a child may be well ahead in the game, expecting to win easily. The child finishes his/her turn on the head of the longest snake and fall dramatically to last place. This type of reversal of fortune can be a rude shock to a young child and can be a valuable and safe early life lesson. It helps to prepare the child for life's little reversals. An added bonus of course is that in Snakes and Ladders you can still win the game even if you're well behind the leader. This is achieved by vaulting other players when you land on one of the tall ladders and then shimmy to the top.

Snakes are at the following positions:

- 36 to 6
- 88 to 24
- 48 to 27

Ladders are at the following positions:

- 4 to 14
- 8 to 30
- 21 to 42
- 28 to 76
- 56 to 95
- 71 to 92
- 78 to 97

Header Files Used

The following header files have been used:

#include<iostream.h>: For input/output functions.

#include<conio.h>: For display functions.

#include<fstream.h>: For file functions.

#include<stdlib.h>: For random number functions.

#include<string.h>: For string functions.

#include<stdio.h>: For input functions.

#include<ctype.h>: For character functions.

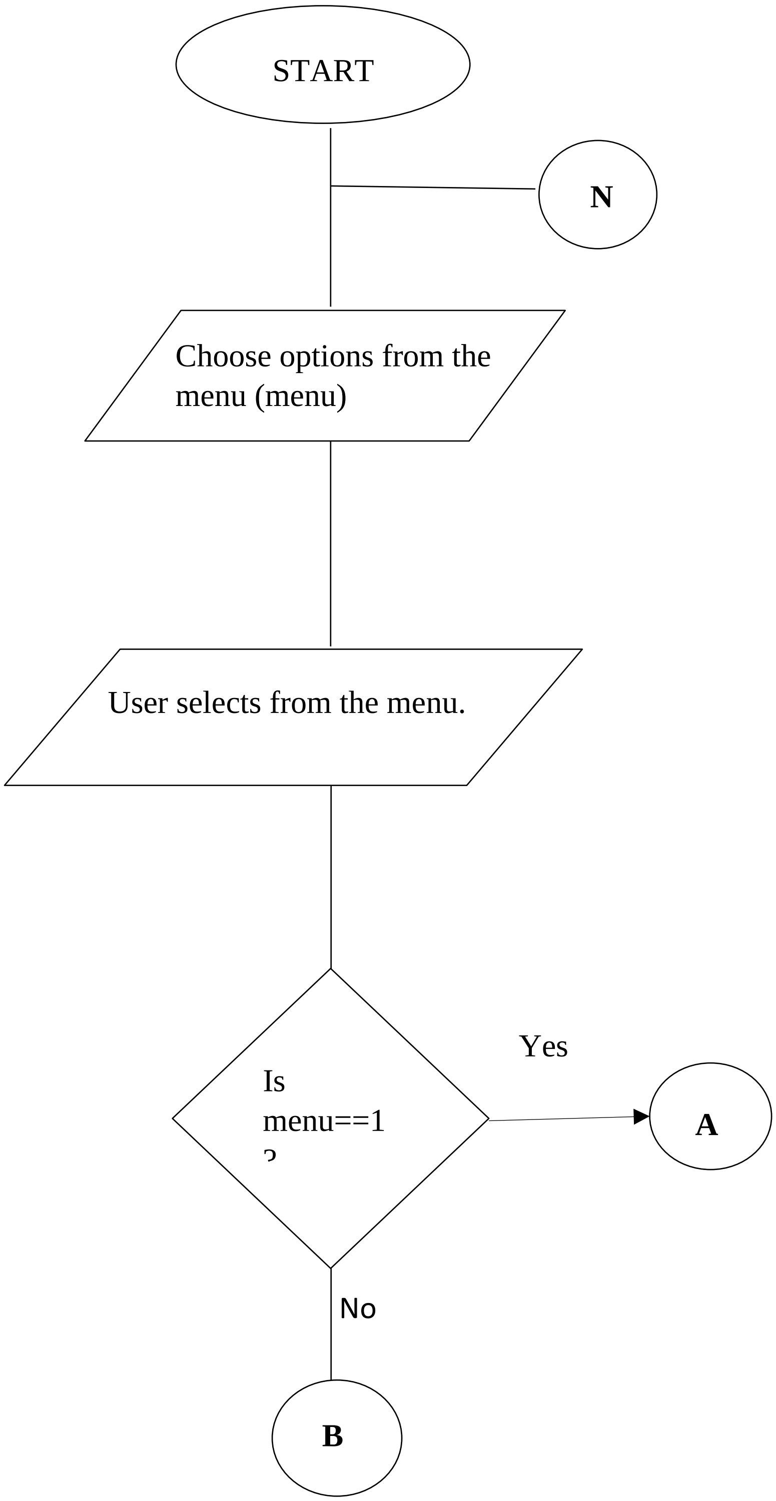
#include<time.h>: For capturing the time for the random function.

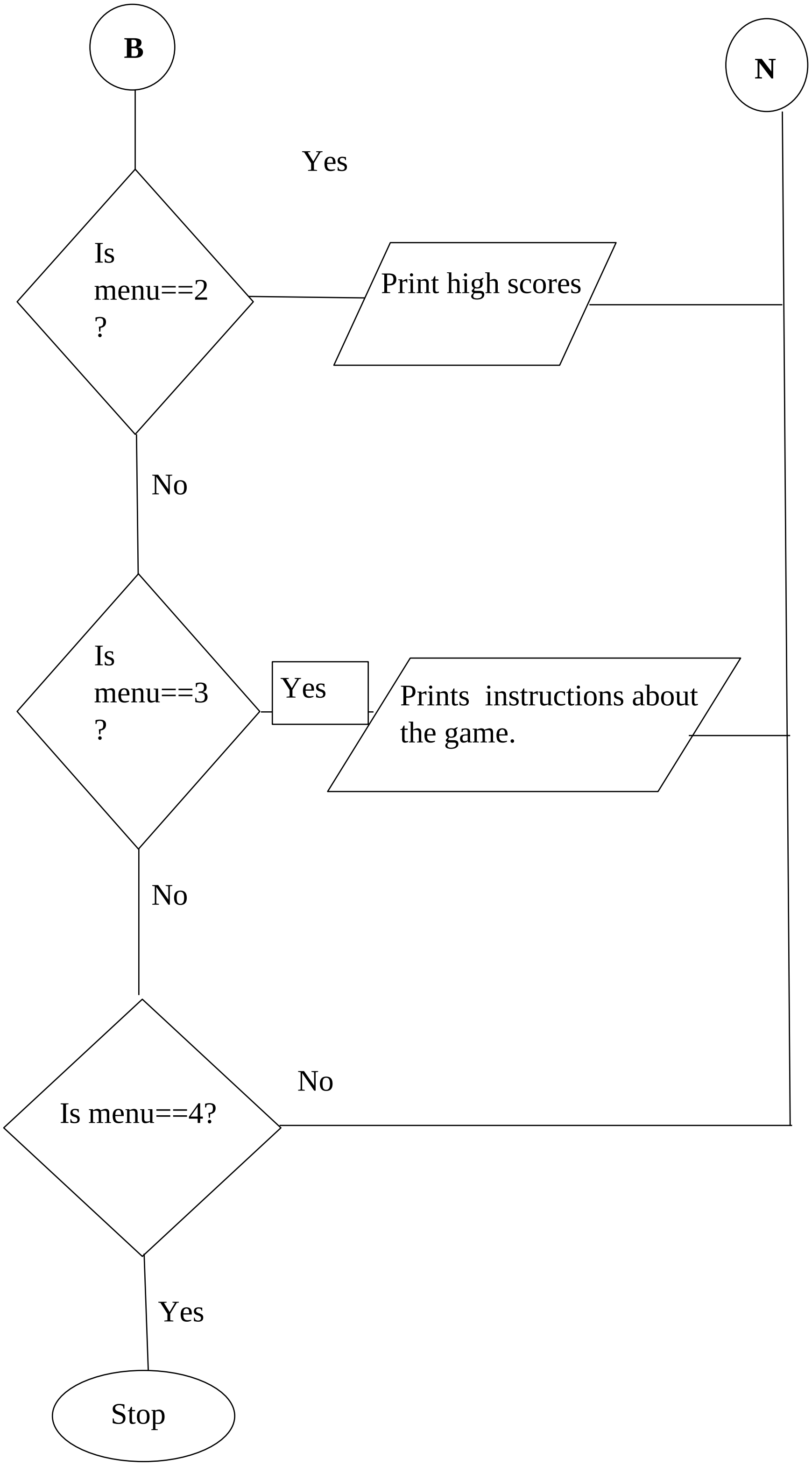
#include<graphics.h>: For graphics.

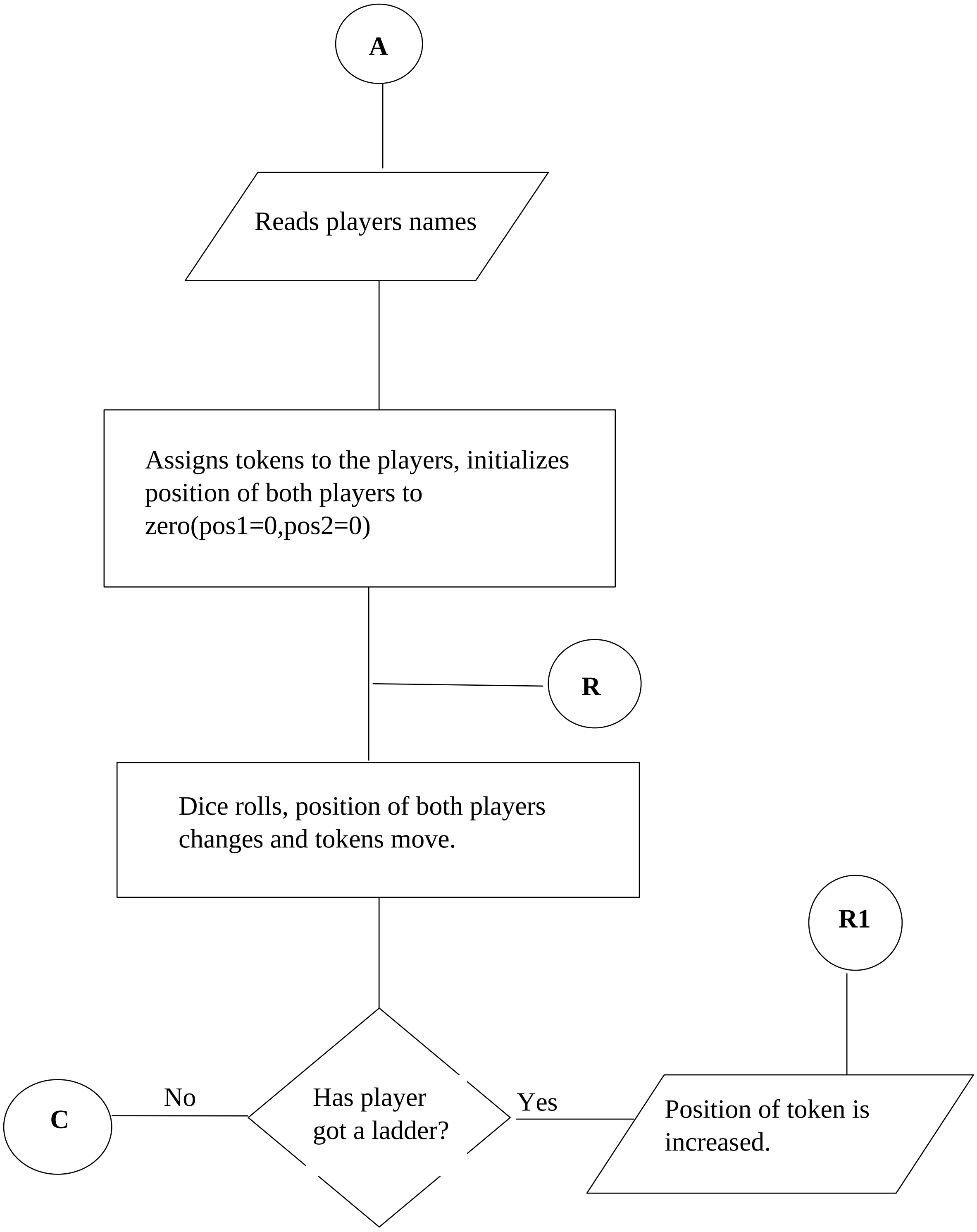
#include<iomanip.h>: For setting the width.

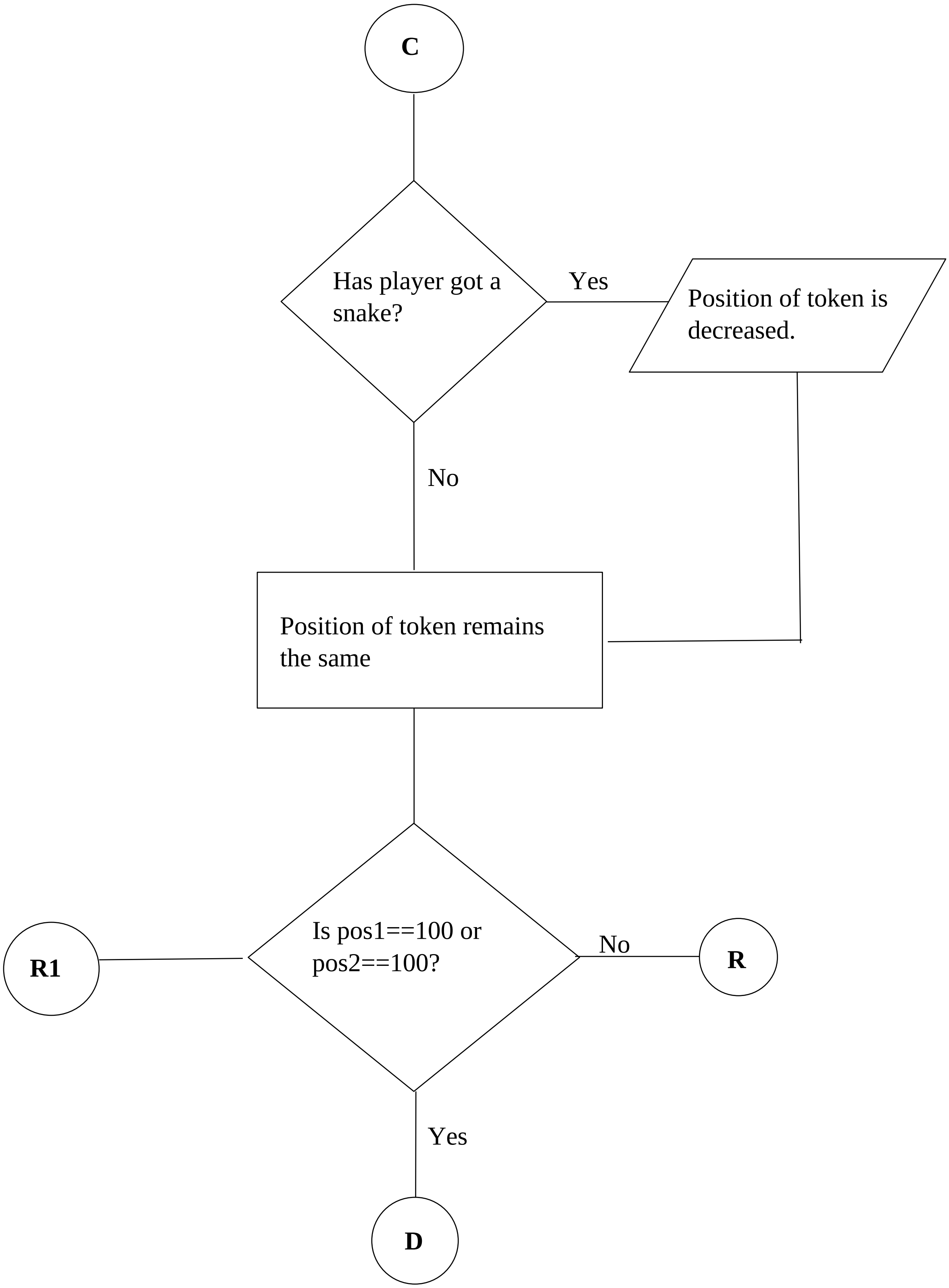
Files Used

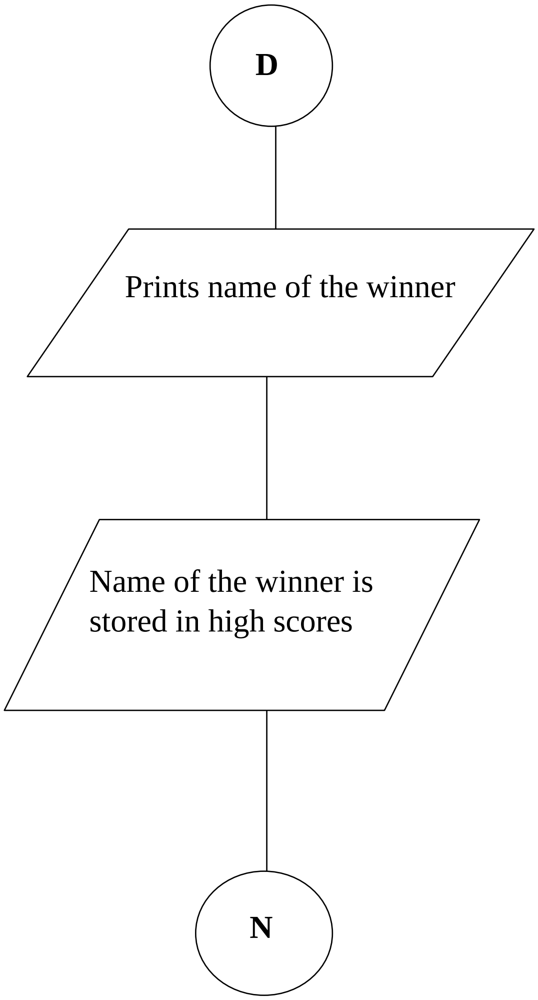
The file "**Details.txt**" is created by the program to store the winner's names.


START
N
Choose options from the menu (menu)
User selects from the menu.
Is menu==1 ?
Yes
A
No
B


B
N
Yes
Is menu==2 ?
Print high scores
No
Is menu==3 ?
Yes
Prints instructions about the game.
No
No
Is menu==4?
Yes
Stop


A
Reads players names
Assigns tokens to the players, initializes position of both players to zero(pos1=0,pos2=0)
R
Dice rolls, position of both players changes and tokens move.
R1
No
Has player got a ladder?
Yes
C
Position of token is increased.


C
Has player got a snake?
Yes
Position of token is decreased.
No
Position of token remains the same
R1
Is pos1==100 or pos2==100?
No
R
Yes
D


D
Prints name of the winner
Name of the winner is stored in high scores
N

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<stdio.h>
#include<graphics.h>
#include<iomanip.h>
#include<fstream.h>
#include<string.h>
#include<ctype.h>
//Global variables for dice and last positions
int n=0,q1=0,q2=0;
//Class for the board of snakes and ladders
class Board
{
        public:
        //Function for the numbers on the board
        void numbering()
        {
                outtextxy(2,373,"1     2     3     4     5     6     7     8     9     10");
                outtextxy(2,333,"20    19    18    17    16    15    14    13    12    11");
                outtextxy(2,293,"21    22    23    24    25    26    27    28    29    30");
                outtextxy(2,253,"40    39    38    37    36    35    34    33    32    31");
                outtextxy(2,213,"41    42    43    44    45    46    47    48    49    50");
                outtextxy(2,173,"60    59    58    57    56    55    54    53    52    51");
                outtextxy(2,133,"61    62    63    64    65    66    67    68    69    70");
                outtextxy(2,93, "80    79    78    77    76    75    74    73    72    71");
                outtextxy(2,53, "81    82    83    84    85    86    87    88    89    90");
                outtextxy(2,13, "100   99    98    97    96    95    94    93    92    91");
        setcolor(15);

    }

    //Function for the lines of ladders on the board

    void ladders()

    {

        setcolor(2);

        line(192.5,373,346,318);//4 to 14

        line(423.5,373,520.5,280);//8 to 30

        line(423.5,285,240,108);//28 to 76

        line(520.5,108,480,13);//71 to 92

        line(38.5,293,77,213);//21 to 42

        line(135,93,192.5,13);//78 to 97

        line(251,173,318,13);//95 to 56

    }

    //Function for the lines of snakes on the board

    void snakes()

    {

        setcolor(12);

        line(423.5,53,192.5,293);//88 to 24

        line(240,253,318,373);//36 to 6

        line(423.5,213,366,293);//48 to 27

    }

};

    void CreateBoard();//Function for the grid

    void dice(int &y);//Function to generate random values for dice

    void play(int &pos,int &q);//Function to give the position of the player

    int luck(int &pos);//Function for snakes and ladders

    void game();//Function to declare the winner

    void menu();//Function for the menu of the game

    void highscores();//Funtion to view the winners of the game from the file
    void ins();//Function for the instructions of the game
    void screen(char name1[35],char name2[35],int& pos1,int& pos2);//Function for the symbols on the board
void main()
{
    menu();
}
void menu()
{
    while(1)
    {
        clrscr();
        //Menu
        cout<<"Welcome to Snakes and Ladders"<<endl;
        cout<<"1. New Game\n";
        cout<<"2. High Scores\n";
        cout<<"3. Instructions\n";
        cout<<"4. Exit\n";
        cout<<"Enter the digit to continue:\n"<<endl;
        //Type casting
        char menu;
        menu=getch();
        if(isdigit(menu))
            {
            if(menu=='1')
                game();
            else if(menu=='2')
                highscores();
            else if(menu=='3')
                ins();
            else if(menu=='4')
                exit(0);
            else
                cout<<"Enter the digits provided from the Menu.";
            getch();
            }
    }
}
void ins()
{
    //Instructions
    cout<<"\n The instructions of the game are:";
    cout<<"\n1.Token of player 1-'@'";
    cout<<"\n2.Token of player 2-'#'";
    cout<<"\n3.Green lines on the board represent ladders and red lines represent snakes";
    cout<<"\n4.Press any key to roll the dice.Dice numbers are randomly generated";
    cout<<"\n5.The first player to reach 100 on the board wins!";
    cout<<"\n                    ALL THE BEST";
}
void game()
{
    clrscr();
    ofstream fout;
    fout.open("details.txt",ios::app);
    char name1[35],name2[35];
    cout<<"\nEnter name of player 1:";
    gets(name1);
    cout<<"\nEnter name of player 2:";
    gets(name2);
    int pos1=0,pos2=0;
```

```
        pos1=pos2;
        while(pos1<=100&&pos2<=100)
        {
                screen(name1,name2,pos1,pos2);
                if(pos1>=100 || pos2>=100)
                break;
        }
        //Declaring and reading the name of the winner to the file
        if (pos1>=100)
        {
                cout<<"\n Congrats! "<<name1<<" you won the game.";
                fout<<name1<<endl;
        }
        else if(pos2>=100)
        {
                cout<<"\n Congrats! "<<name2<<" you won the game.";
                fout<<name2<<endl;
        }
        fout.close();
        getch();
}
void highscores()
{
        clrscr();
        char *ch;
        ifstream fin;
        fin.open("details.txt",ios::in);
        //Printing the names of the winners from the file
        while(!fin.eof())
        {
}               fin>>ch;
                cout<<ch<<endl;
        }
        fin.close();
        getch();
        menu();
}
void dice(int &y)
{
        //Random number for the dice
        srand(time(NULL));
        n=rand()%6;
        getch();
        y=y+n+1;
        if(y>100)
        y=y-n-1;
}
void play(int &pos,int &q)
{       //Position of the winner
        int lastpos=pos;
        dice(pos);
        lastpos=luck(pos);
        if(luck(pos)>pos)
                cout<<"\nCongrats! You got a ladder and moved to "<<luck(pos);
        else if(luck(pos)<pos)
                cout<<"\nOops! You got a snake and moved to "<<luck(pos);
        cout<<"\n---------------You are at position: ";
        pos=lastpos;
        q=pos;
}
```

```
int luck(int &pos)
{
        switch(pos)
        {
                //Ladders
                case 4:
                        return 14;
                case 8:
                        return 30;
                case 21:
                        return 42;
                case 28:
                        return 76;
                case 71:
                        return 92;
                case 78:
                        return 97;
                case 56:
                        return 95;
                //Snakes
                case 88:
                        return 24;
                case 36:
                        return 6;
                case 48:
                        return 27;
                default:
                        return pos;
        }
}
```

```
void CreateBoard()
{       //Arrangement of Columns of the Grid
            float x1=0,y1=0,x2=0,y2=385;
            for(int i=0;i<11;i++)
            {
                        line(x1,y1,x2,y2);
                        x1+=56;
                        x2+=56;
            }
            //Arrangement of Rows of the Grid
            x1=0,y1=0,x2=560,y2=0;
            for(i=0;i<11;i++)
            {
                        line(x1,y1,x2,y2);
                        y1+=38.5;
                        y2+=38.5;
            }
}
void screen(char name1[35],char name2[35],int& pos1,int& pos2)
{
            clrscr();
            //Opening of Graphics mode
            int gdriver= DETECT,gmode,xp,yp,xp1,yp1;
            initgraph(&gdriver,&gmode,"c:\\turboc3\\bgi");
            Board b;
            CreateBoard();
            b.numbering();
            b.ladders();
            b.snakes();
            //Changing the names of player 1 and player 2 to uppercase
```

```
for(int i=0;name1[i]!='\0';i++)
        name1[i]=toupper(name1[i]);
for( i=0;name2[i]!='\0';i++)
        name2[i]=toupper(name2[i]);
cout<<"\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n";
cout<<endl<<name1<<" your turn (last position "<<q1<<"). Enter a key";
play(pos1,q1);
//Symbol for player 1
switch(pos1)
{
        case 1:
                xp=25;
                yp=373;
                break;
        case 2:
                xp=85;
                yp=373;
                break;
        case 3:
                xp=145;
                yp=373;
                break;
        case 4:
                xp=204;
                yp=373;
                break;
        case 5:
                xp=255;
                yp=373;
                break;
```

```
case 6:
        xp=305;
        yp=373;
        break;
case 7:
        xp=365;
        yp=373;
        break;
case 8:
        xp=425;
        yp=373;
        break;
case 9:
        xp=485;
        yp=373;
        break;
case 10:
        xp=540;
        yp=373;
        break;
case 11:
        xp=540;
        yp=333;
        break;
case 12:
        xp=485;
        yp=333;
        break;
case 13:
        xp=425;
        yp=333;
        break;
case 14:
        xp=365;
        yp=333;
        break;
case 15:
        xp=305;
        yp=333;
        break;
case 16:
        xp=255;
        yp=333;
        break;
case 17:
        xp=204;
        yp=333;
        break;
case 18:
        xp=145;
        yp=333;
        break;
case 19:
        xp=85;
        yp=333;
        break;
case 20:
        xp=25;
        yp=333;
        break;
```

```
case 21:
        xp=25;
        yp=293;
        break;
case 22:
        xp=85;
        yp=293;
        break;
case 23:
        xp=145;
        yp=293;
        break;
case 24:
        xp=204;
        yp=293;
        break;
case 25:
        xp=255;
        yp=293;
        break;
case 26:
        xp=305;
        yp=293;
        break;
case 27:
        xp=365;
        yp=293;
        break;
case 28:
        xp=425;
        yp=293;
        break;
case 29:
        xp=485;
        yp=293;
        break;
case 30:
        xp=540;
        yp=293;
        break;
case 31:
        xp=540;
        yp=253;
        break;
case 32:
        xp=485;
        yp=253;
        break;
case 33:
        xp=425;
        yp=253;
        break;
case 34:
        xp=365;
        yp=253;
        break;
case 35:
        xp=305;
        yp=253;
        break;
```

```
case 36:
        xp=255;
        yp=253;
        break;
case 37:
        xp=204;
        yp=253;
        break;
case 38:
        xp=145;
        yp=253;
        break;
case 39:
        xp=85;
        yp=253;
        break;
case 40:
        xp=25;
        yp=253;
        break;
case 41:
        xp=25;
        yp=213;
        break;
case 42:
        xp=85;
        yp=213;
        break;
case 43:
        xp=145;
```

```
                yp=213;
                break;
case 44:
                xp=204;
                yp=213;
                break;
case 45:
                xp=255;
                yp=213;
                break;
case 46:
                xp=305;
                yp=213;
                break;
case 47:
                xp=365;
                yp=213;
                break;
case 48:
                xp=425;
                yp=213;
                break;
case 49:
                xp=485;
                yp=213;
                break;
case 50:
                xp=540;
                yp=213;
                break;
```

```
case 51:
        xp=540;
        yp=173;
        break;
case 52:
        xp=485;
        yp=173;
        break;
case 53:
        xp=425;
        yp=173;
        break;
case 54:
        xp=365;
        yp=173;
        break;
case 55:
        xp=305;
        yp=173;
        break;
case 56:
        xp=255;
        yp=173;
        break;
case 57:
        xp=204;
        yp=173;
        break;
case 58:
        xp=145;
```

```
                yp=173;
                break;
        case 59:
                xp=85;
                yp=173;
                break;
        case 60:
                xp=25;
                yp=173;
                break;
        case 61:
                xp=25;
                yp=133;
                break;
        case 62:
                xp=85;
                yp=133;
                break;
        case 63:
                xp=145;
                yp=133;
                break;
        case 64:
                xp=204;
                yp=133;
                break;
        case 65:
                xp=255;
                yp=133;
                break;
```

```
case 66:
        xp=305;
        yp=133;
        break;
case 67:
        xp=365;
        yp=133;
        break;
case 68:
        xp=425;
        yp=133;
        break;
case 69:
        xp=485;
        yp=133;
        break;
case 70:
        xp=540;
        yp=133;
        break;
case 71:
        xp=540;
        yp=93;
        break;
case 72:
        xp=485;
        yp=93;
        break;
case 73:
        xp=425;
        yp=93;
        break;
case 74:
        xp=365;
        yp=93;
        break;
case 75:
        xp=305;
        yp=93;
        break;
case 76:
        xp=255;
        yp=93;
        break;
case 77:
        xp=204;
        yp=93;
        break;
case 78:
        xp=145;
        yp=93;
        break;
case 79:
        xp=85;
        yp=93;
        break;
case 80:
        xp=25;
        yp=93;
        break;
```

```
case 81:
            xp=25;
            yp=53;
            break;
case 82:
            xp=85;
            yp=53;
            break;
case 83:
            xp=145;
            yp=53;
            break;
case 84:
            xp=204;
            yp=53;
            break;
case 85:
            xp=255;
            yp=53;
            break;
case 86:
            xp=305;
            yp=53;
            break;
case 87:
            xp=365;
            yp=53;
            break;
case 88:
            xp=425;
```

```
        yp=53;
        break;
case 89:
        xp=485;
        yp=53;
        break;
case 90:
        xp=540;
        yp=53;
        break;
case 91:
        xp=540;
        yp=13;
        break;
case 92:
        xp=485;
        yp=13;
        break;
case 93:
        xp=425;
        yp=13;
        break;
case 94:
        xp=365;
        yp=13;
        break;
case 95:
        xp=305;
        yp=13;
        break;
        case 96:
                xp=255;
                yp=13;
                break;
        case 97:
                xp=204;
                yp=13;
                break;
        case 98:
                xp=145;
                yp=13;
                break;
        case 99:
                xp=85;
                yp=13;
                break;
        case 100:
                xp=25;
                yp=13;
                break;
        }

cout<<" "<<pos1<<"-----------------\t\tDice:"<<n+1;
setcolor(5);//Colour of the symbol for player 1 - Purple
outtextxy(xp,yp,"@");
getch();
cout<<"\n"<<name2<<" your turn(last position "<<q2<<"). Enter a key";
play(pos2,q2);
//Symbol for player 2
switch(pos2)
```

```
{
        case 1:
                xp1=25;
                yp1=373;
                break;
        case 2:
                xp1=85;
                yp1=373;
                break;
        case 3:
                xp1=145;
                yp1=373;
                break;
        case 4:
                xp1=204;
                yp1=373;
                break;
        case 5:
                xp1=255;
                yp1=373;
                break;
        case 6:
                xp1=305;
                yp1=373;
                break;
        case 7:
                xp1=365;
                yp1=373;
                break;
        case 8:
            xp1=425;
            yp1=373;
            break;
case 9:
            xp1=485;
            yp1=373;
            break;
case 10:
            xp1=540;
            yp1=373;
            break;
case 11:
            xp1=540;
            yp1=333;
            break;
case 12:
            xp1=485;
            yp1=333;
            break;
case 13:
            xp1=425;
            yp1=333;
            break;
case 14:
            xp1=365;
            yp1=333;
            break;
case 15:
            xp1=305;
            yp1=333;
            break;
case 16:
            xp1=255;
            yp1=333;
            break;
case 17:
            xp1=204;
            yp1=333;
            break;
case 18:
            xp1=145;
            yp1=333;
            break;
case 19:
            xp1=85;
            yp1=333;
            break;
case 20:
            xp1=25;
            yp1=333;
            break;
case 21:
            xp1=25;
            yp1=293;
            break;
case 22:
            xp1=85;
            yp1=293;
            break;
case 23:
```

```
                xp1=145;
                yp1=293;
                break;
case 24:
                xp1=204;
                yp1=293;
                break;
case 25:
                xp1=255;
                yp1=293;
                break;
case 26:
                xp1=305;
                yp1=293;
                break;
case 27:
                xp1=365;
                yp1=293;
                break;
case 28:
                xp1=425;
                yp1=293;
                break;
case 29:
                xp1=485;
                yp1=293;
                break;
case 30:
                xp1=540;
                yp1=293;
```

```
        break;
case 31:
        xp1=540;
        yp1=253;
        break;
case 32:
        xp1=485;
        yp1=253;
        break;
case 33:
        xp1=425;
        yp1=253;
        break;
case 34:
        xp1=365;
        yp1=253;
        break;
case 35:
        xp1=305;
        yp1=253;
        break;
case 36:
        xp1=255;
        yp1=253;
        break;
case 37:
        xp1=204;
        yp1=253;
        break;
case 38:
        xp1=145;
        yp1=253;
        break;
case 39:
        xp1=85;
        yp1=253;
        break;
case 40:
        xp1=25;
        yp1=253;
        break;
case 41:
        xp1=25;
        yp1=213;
        break;
case 42:
        xp1=85;
        yp1=213;
        break;
case 43:
        xp1=145;
        yp1=213;
        break;
case 44:
        xp1=204;
        yp1=213;
        break;
case 45:
        xp1=255;
        yp1=213;
```

```
            break;
case 46:
            xp1=305;
            yp1=213;
            break;
case 47:
            xp1=365;
            yp1=213;
            break;
case 48:
            xp1=425;
            yp1=213;
            break;
case 49:
            xp1=485;
            yp1=213;
            break;
case 50:
            xp1=540;
            yp1=213;
            break;
case 51:
            xp1=540;
            yp1=173;
            break;
case 52:
            xp1=485;
            yp1=173;
            break;
case 53:
```

```
            xp1=425;
            yp1=173;
            break;
case 54:
            xp1=365;
            yp1=173;
            break;
case 55:
            xp1=305;
            yp1=173;
            break;
case 56:
            xp1=255;
            yp1=173;
            break;
case 57:
            xp1=204;
            yp1=173;
            break;
case 58:
            xp1=145;
            yp1=173;
            break;
case 59:
            xp1=85;
            yp1=173;
            break;
case 60:
            xp1=25;
            yp1=173;
```

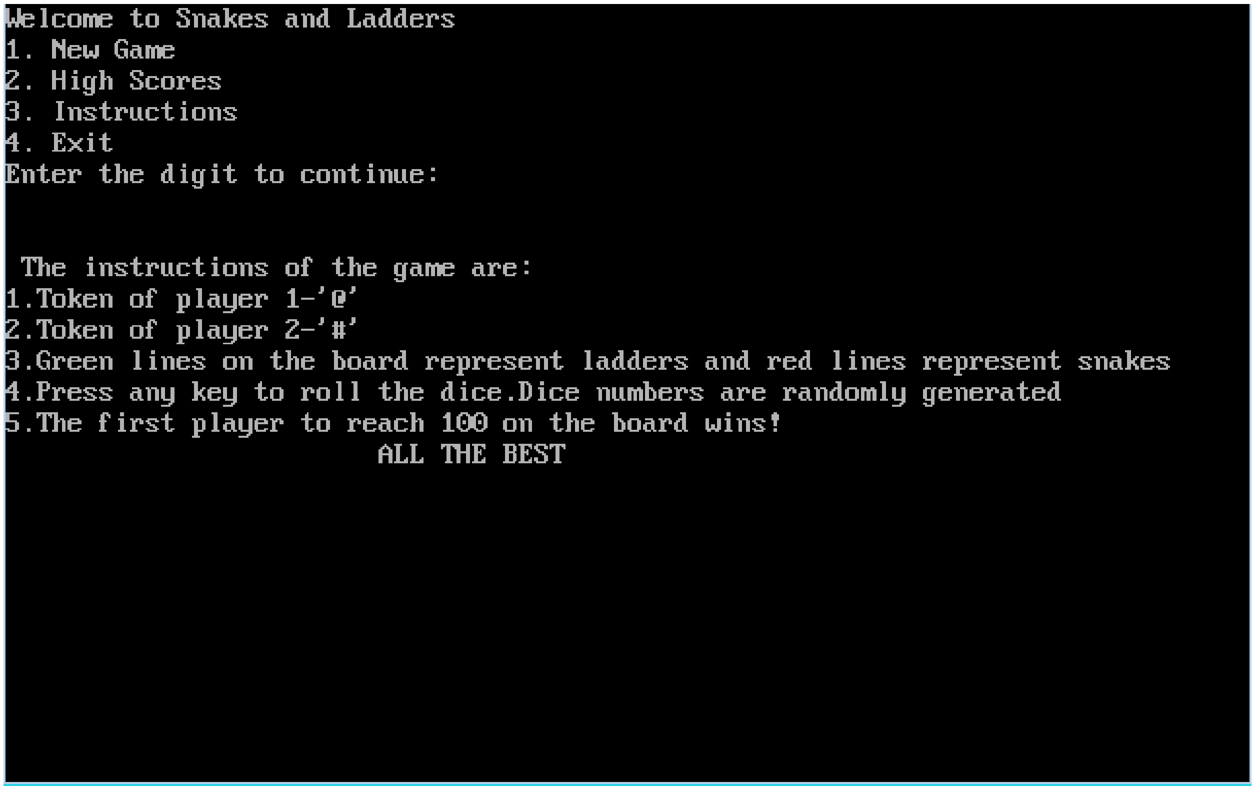
```
        break;
case 61:
        xp1=25;
        yp1=133;
        break;
case 62:
        xp1=85;
        yp1=133;
        break;
case 63:
        xp1=145;
        yp1=133;
        break;
case 64:
        xp1=204;
        yp1=133;
        break;
case 65:
        xp1=255;
        yp1=133;
        break;
case 66:
        xp1=305;
        yp1=133;
        break;
case 67:
        xp1=365;
        yp1=133;
        break;
case 68:
        xp1=425;
        yp1=133;
        break;
case 69:
        xp1=485;
        yp1=133;
        break;
case 70:
        xp1=540;
        yp1=133;
        break;
case 71:
        xp1=540;
        yp1=93;
        break;
case 72:
        xp1=485;
        yp1=93;
        break;
case 73:
        xp1=425;
        yp1=93;
        break;
case 74:
        xp1=365;
        yp1=93;
        break;
case 75:
        xp1=305;
        yp1=93;
        break;
case 76:
        xp1=255;
        yp1=93;
        break;
case 77:
        xp1=204;
        yp1=93;
        break;
case 78:
        xp1=145;
        yp1=93;
        break;
case 79:
        xp1=85;
        yp1=93;
        break;
case 80:
        xp1=25;
        yp1=93;
        break;
case 81:
        xp1=25;
        yp1=53;
        break;
case 82:
        xp1=85;
        yp1=53;
        break;
case 83:
```

```
            xp1=145;
            yp1=53;
            break;
    case 84:
            xp1=204;
            yp1=53;
            break;
    case 85:
            xp1=255;
            yp1=53;
            break;
    case 86:
            xp1=305;
            yp1=53;
            break;
    case 87:
            xp1=365;
            yp1=53;
            break;
    case 88:
            xp1=425;
            yp1=53;
            break;
    case 89:
            xp1=485;
            yp1=53;
            break;
    case 90:
            xp1=540;
            yp1=53;
```

```
        break;
case 91:
        xp1=540;
        yp1=13;
        break;
case 92:
        xp1=485;
        yp1=13;
        break;
case 93:
        xp1=425;
        yp1=13;
        break;
case 94:
        xp1=365;
        yp1=13;
        break;
case 95:
        xp1=305;
        yp1=13;
        break;
case 96:
        xp1=255;
        yp1=13;
        break;
case 97:
        xp1=204;
        yp1=13;
        break;
case 98:
```

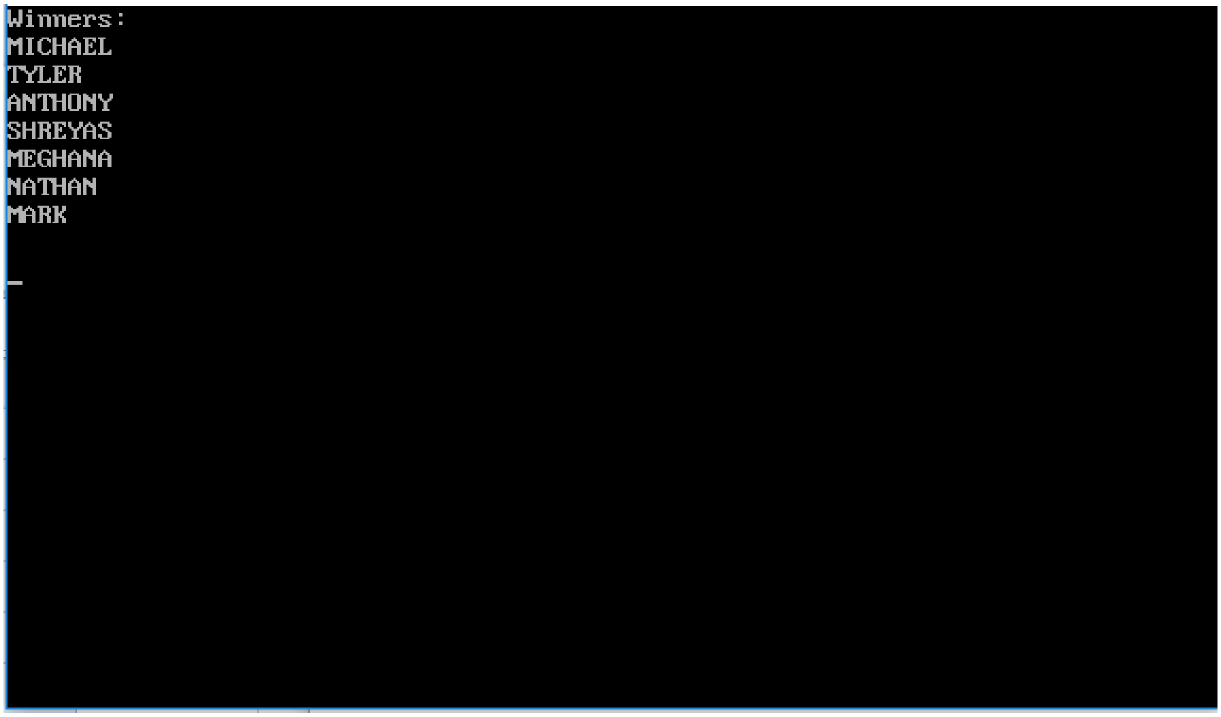
```
                    xp1=145;
                    yp1=13;
                    break;
            case 99:
                    xp1=85;
                    yp1=13;
                    break;
            case 100:
                    xp1=25;
                    yp1=13;
                    break;
            }
        cout<<" "<<pos2<<"-----------------\t\tDice:"<<n+1;
        setcolor(14);//Colour of the symbol for player 2 - Yellow
        outtextxy(xp1,yp1,"#");
        getch();
        closegraph();
}
```

Welcome & Instruction Screen



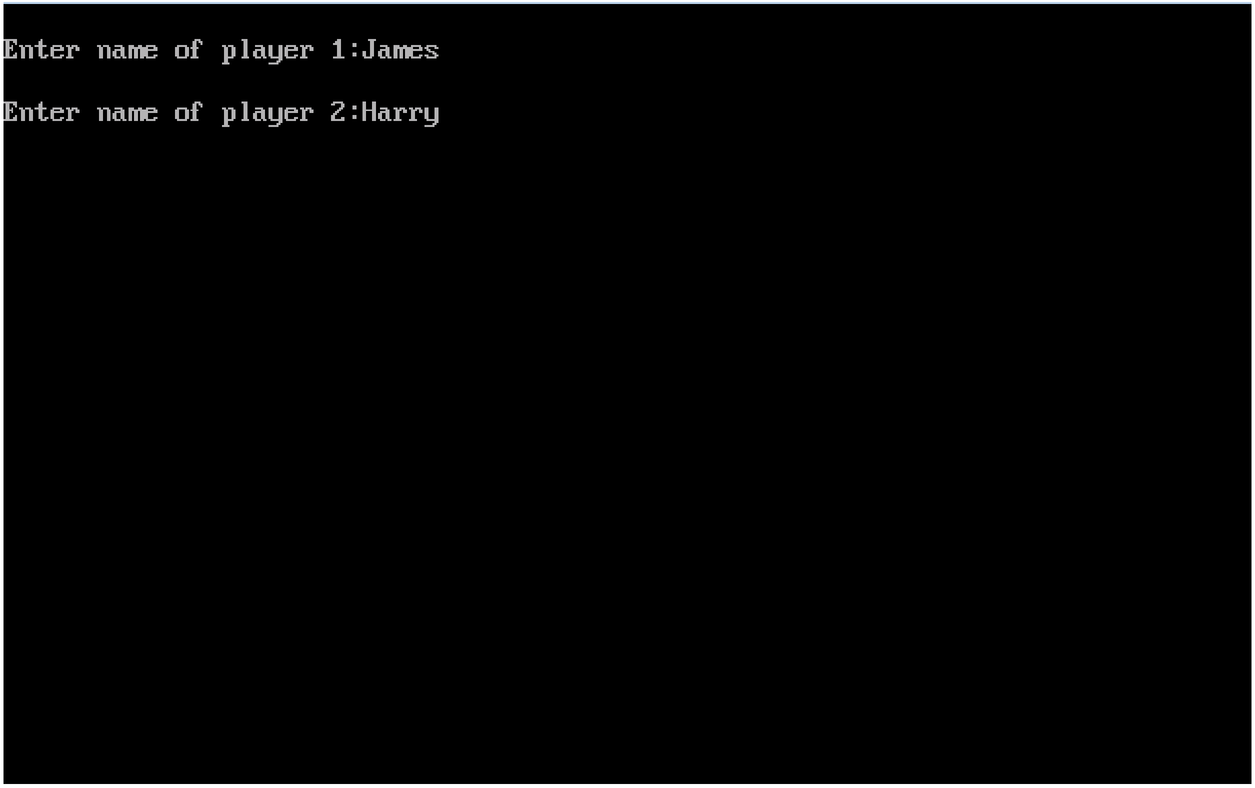
```
Welcome to Snakes and Ladders
1. New Game
2. High Scores
3. Instructions
4. Exit
Enter the digit to continue:


 The instructions of the game are:
1.Token of player 1-'@'
2.Token of player 2-'#'
3.Green lines on the board represent ladders and red lines represent snakes
4.Press any key to roll the dice.Dice numbers are randomly generated
5.The first player to reach 100 on the board wins!
                        ALL THE BEST
```

High scores before the game

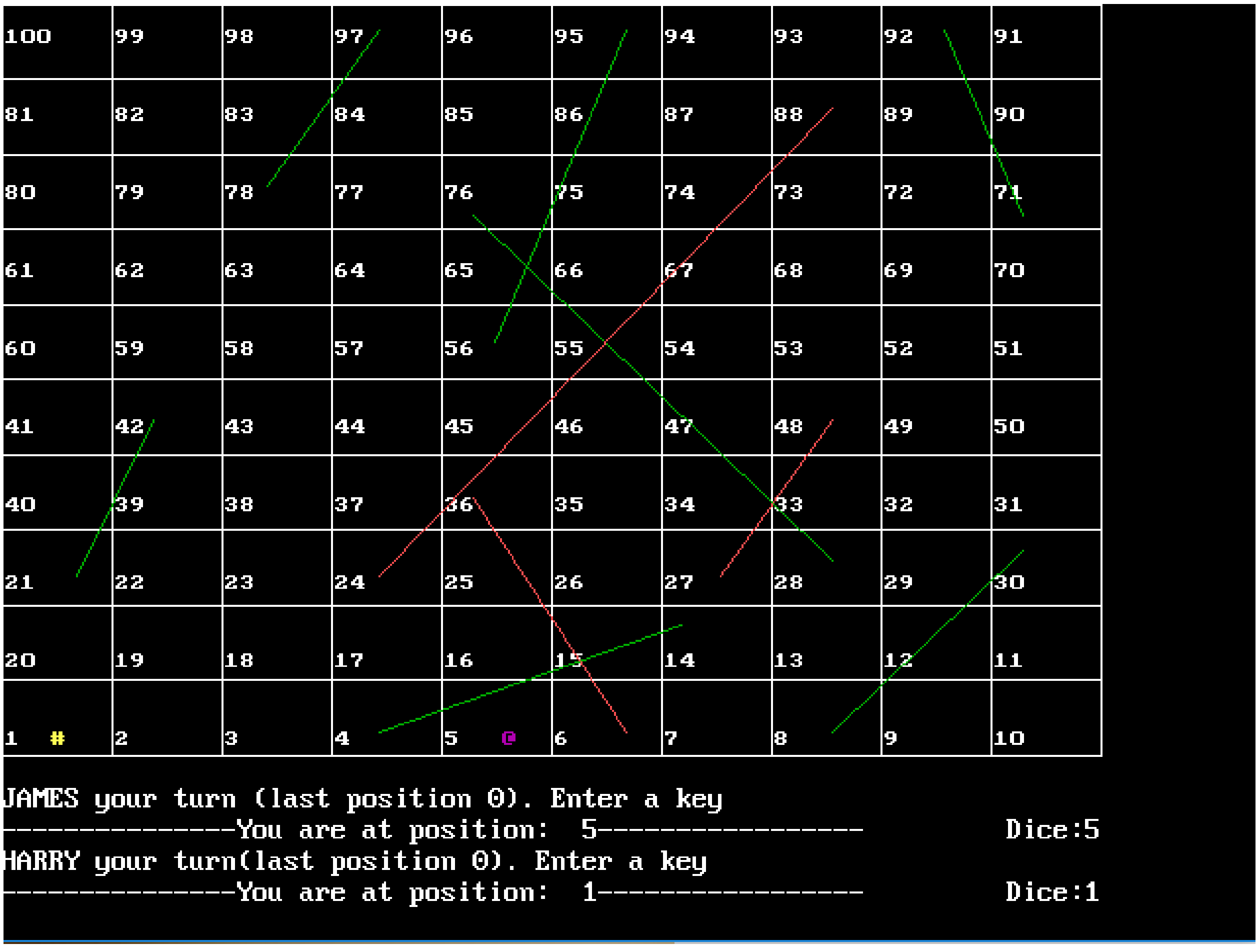


```
Winners:
MICHAEL
TYLER
ANTHONY
SHREYAS
MEGHANA
NATHAN
MARK

_
```

Players Entering their names



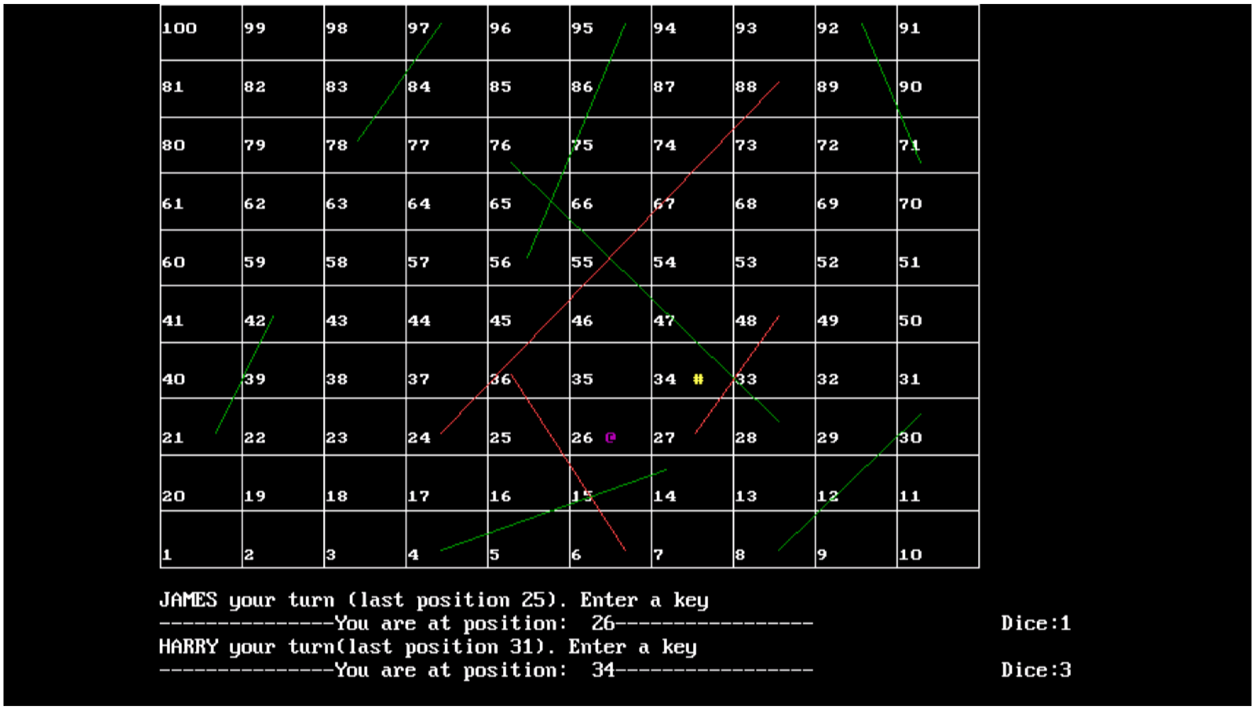
```
Enter name of player 1:James

Enter name of player 2:Harry
```



100	99	98	97	96	95	94	93	92	91
81	82	83	84	85	86	87	88	89	90
80	79	78	77	76	75	74	73	72	71
61	62	63	64	65	66	67	68	69	70
60	59	58	57	56	55	54	53	52	51
41	42	43	44	45	46	47	48	49	50
40	39	38	37	36	35	34	33	32	31
21	22	23	24	25	26	27	28	29	30
20	19	18	17	16	15	14	13	12	11
1 #	2	3	4	5 @	6	7	8	9	10

```
JAMES your turn (last position 0). Enter a key
---------------You are at position:  5-----------------          Dice:5
HARRY your turn(last position 0). Enter a key
---------------You are at position:  1-----------------          Dice:1
```

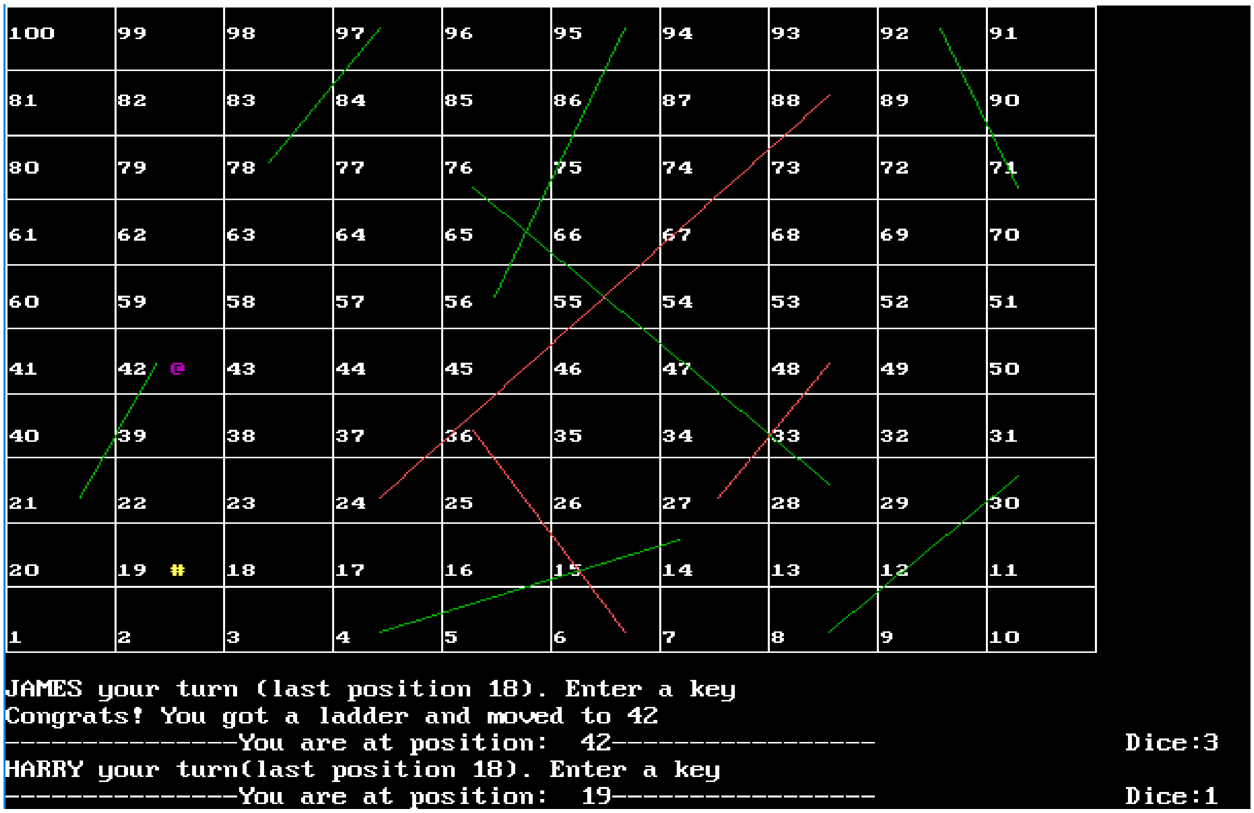
Game in progress



100	99	98	97	96	95	94	93	92	91
81	82	83	84	85	86	87	88	89	90
80	79	78	77	76	75	74	73	72	71
61	62	63	64	65	66	67	68	69	70
60	59	58	57	56	55	54	53	52	51
41	42	43	44	45	46	47	48	49	50
40	39	38	37	36	35	34 #	33	32	31
21	22	23	24	25	26 @	27	28	29	30
20	19	18	17	16	15	14	13	12	11
1	2	3	4	5	6	7	8	9	10

```
JAMES your turn (last position 25). Enter a key
---------------You are at position:  26-----------------          Dice:1
HARRY your turn(last position 31). Enter a key
---------------You are at position:  34-----------------          Dice:3
```

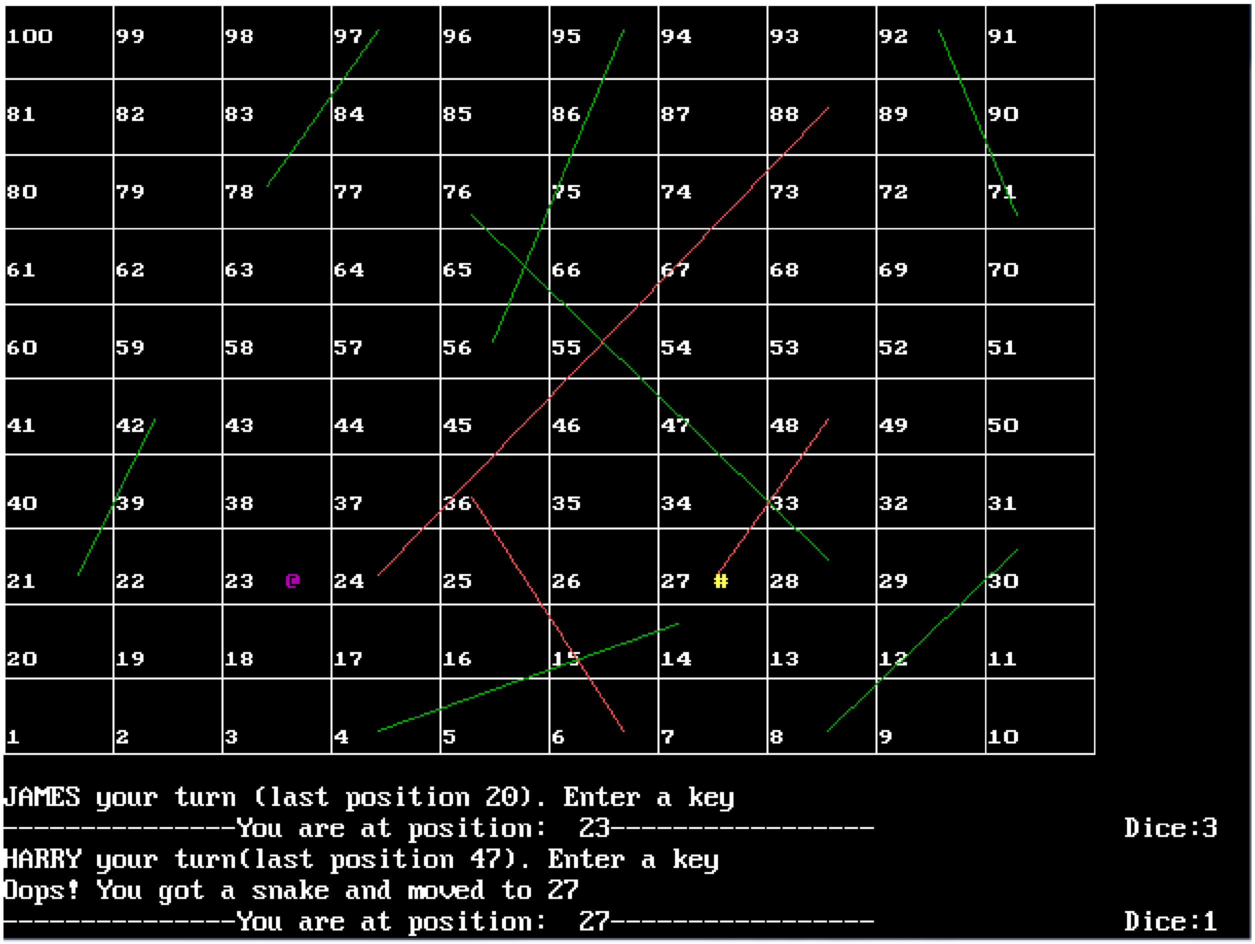
Player gets a ladder



100	99	98	97	96	95	94	93	92	91
81	82	83	84	85	86	87	88	89	90
80	79	78	77	76	75	74	73	72	71
61	62	63	64	65	66	67	68	69	70
60	59	58	57	56	55	54	53	52	51
41	42 @	43	44	45	46	47	48	49	50
40	39	38	37	36	35	34	33	32	31
21	22	23	24	25	26	27	28	29	30
20	19 #	18	17	16	15	14	13	12	11
1	2	3	4	5	6	7	8	9	10

```
JAMES your turn (last position 18). Enter a key
Congrats! You got a ladder and moved to 42
---------------You are at position:  42-----------------          Dice:3
HARRY your turn(last position 18). Enter a key
---------------You are at position:  19-----------------          Dice:1
```

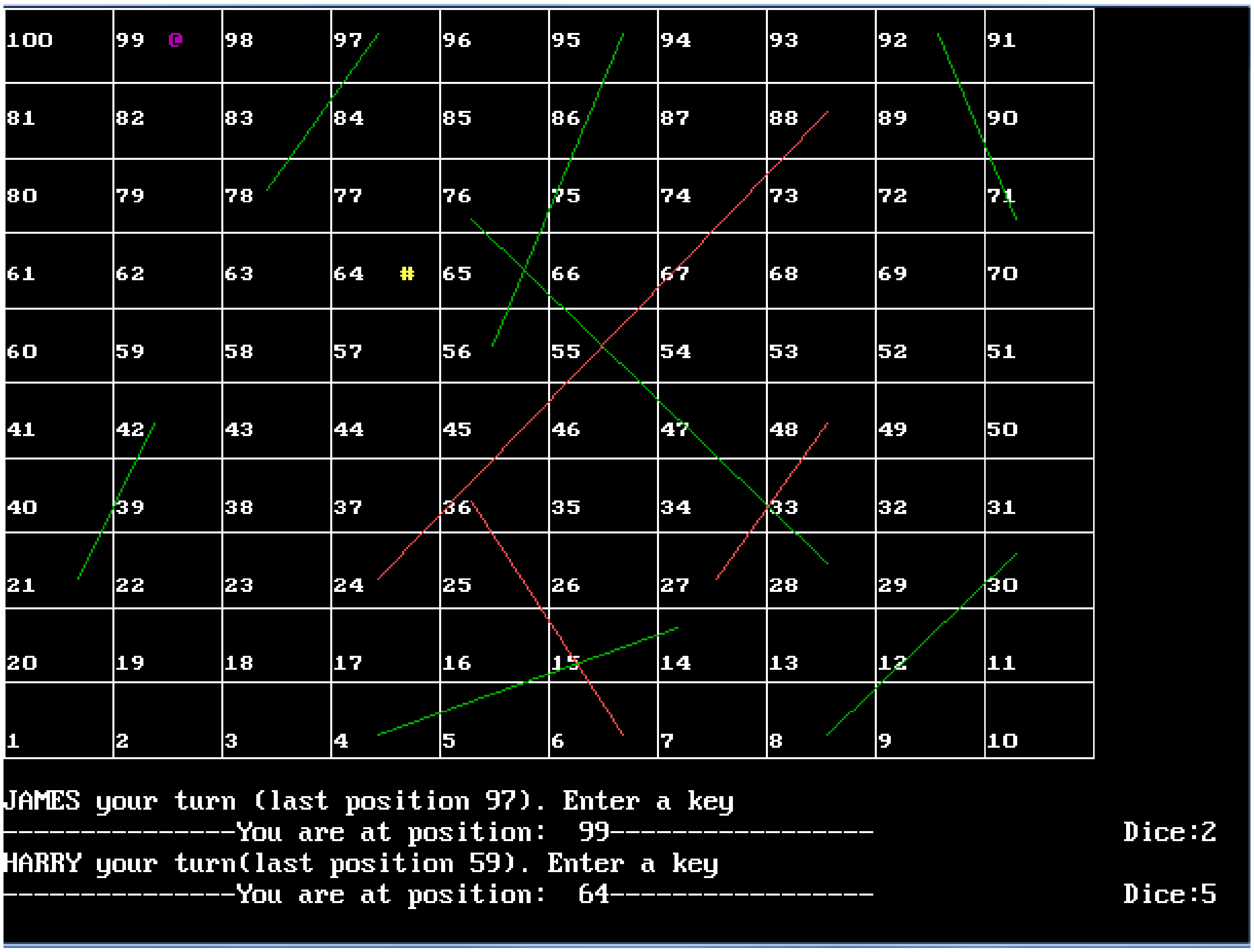
Player gets a snake



100	99	98	97	96	95	94	93	92	91
81	82	83	84	85	86	87	88	89	90
80	79	78	77	76	75	74	73	72	71
61	62	63	64	65	66	67	68	69	70
60	59	58	57	56	55	54	53	52	51
41	42	43	44	45	46	47	48	49	50
40	39	38	37	36	35	34	33	32	31
21	22	23 @	24	25	26	27 #	28	29	30
20	19	18	17	16	15	14	13	12	11
1	2	3	4	5	6	7	8	9	10

```
JAMES your turn (last position 20). Enter a key
---------------You are at position:  23-----------------               Dice:3
HARRY your turn(last position 47). Enter a key
Oops! You got a snake and moved to 27
---------------You are at position:  27-----------------               Dice:1
```

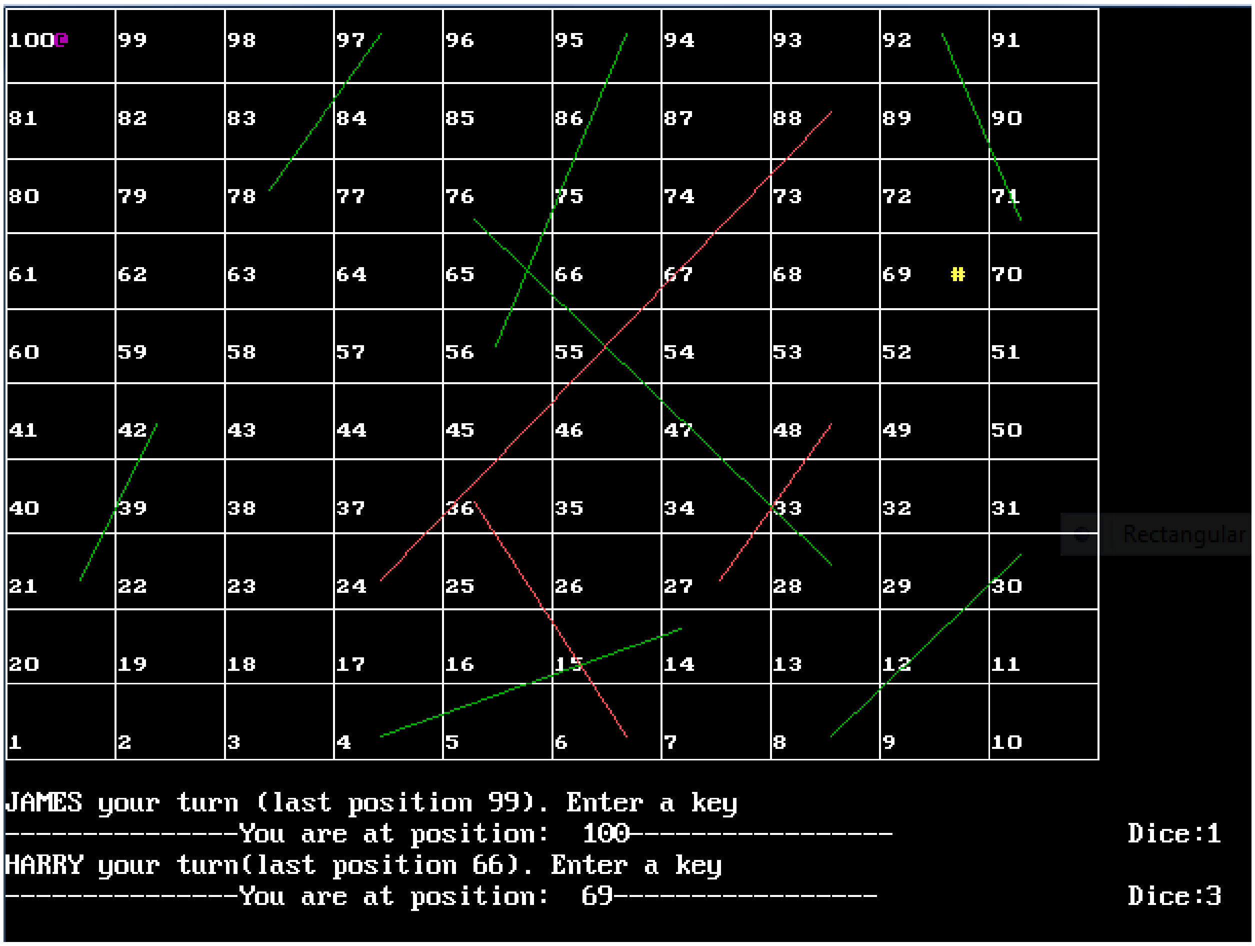
When player’s spaces are limited



100	99 @	98	97	96	95	94	93	92	91
81	82	83	84	85	86	87	88	89	90
80	79	78	77	76	75	74	73	72	71
61	62	63	64 #	65	66	67	68	69	70
60	59	58	57	56	55	54	53	52	51
41	42	43	44	45	46	47	48	49	50
40	39	38	37	36	35	34	33	32	31
21	22	23	24	25	26	27	28	29	30
20	19	18	17	16	15	14	13	12	11
1	2	3	4	5	6	7	8	9	10

```
JAMES your turn (last position 97). Enter a key
---------------You are at position:  99-----------------               Dice:2
HARRY your turn(last position 59). Enter a key
---------------You are at position:  64-----------------               Dice:5
```

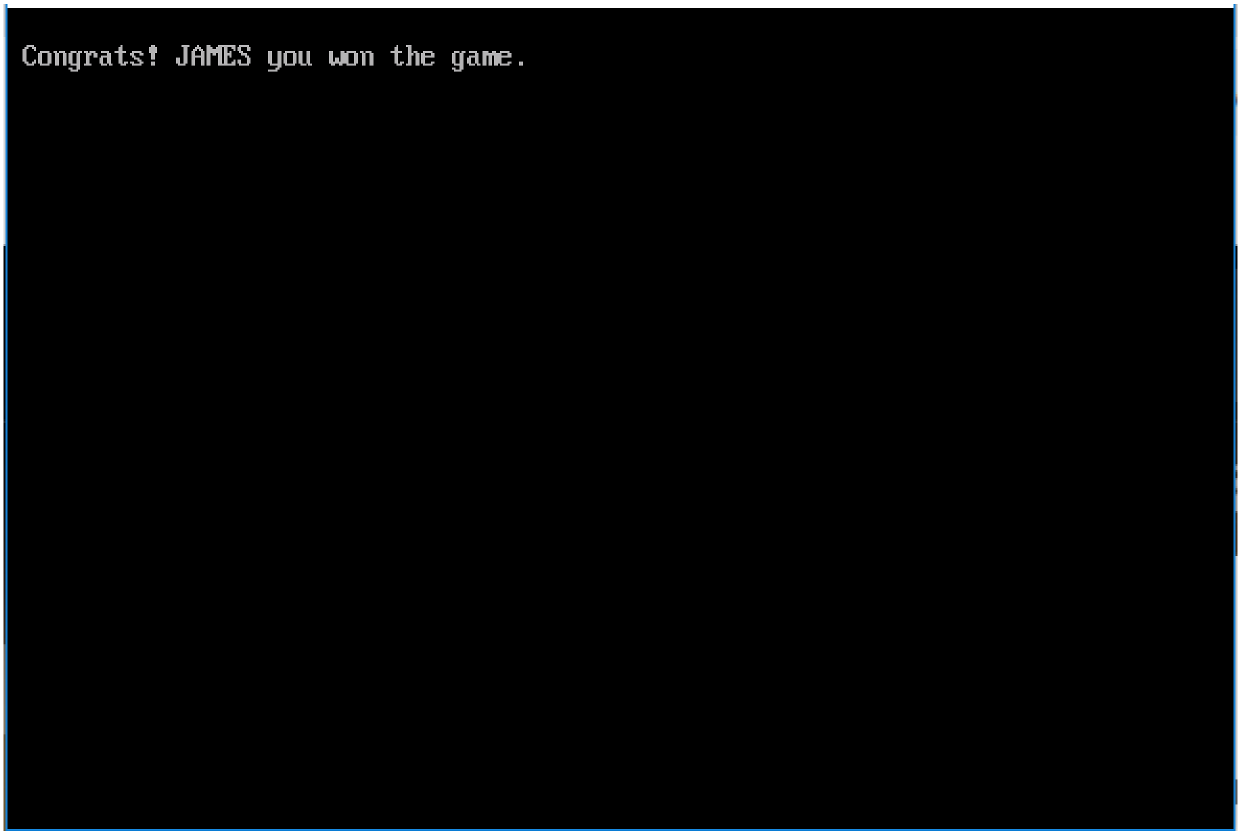
Player on the 100th position



100@	99	98	97	96	95	94	93	92	91
81	82	83	84	85	86	87	88	89	90
80	79	78	77	76	75	74	73	72	71
61	62	63	64	65	66	67	68	69 #	70
60	59	58	57	56	55	54	53	52	51
41	42	43	44	45	46	47	48	49	50
40	39	38	37	36	35	34	33	32	31
21	22	23	24	25	26	27	28	29	30
20	19	18	17	16	15	14	13	12	11
1	2	3	4	5	6	7	8	9	10

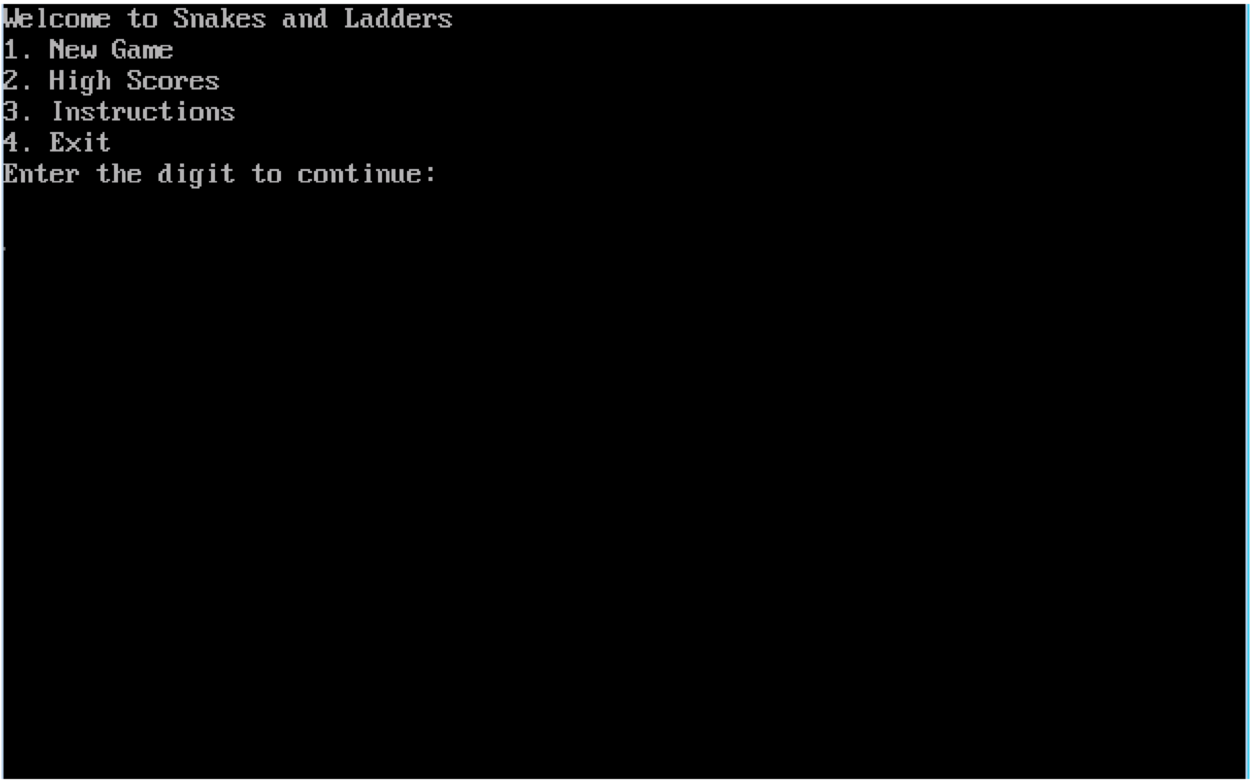
```
JAMES your turn (last position 99). Enter a key
---------------You are at position:  100-----------------              Dice:1
HARRY your turn(last position 66). Enter a key
---------------You are at position:  69-----------------               Dice:3
```

Player winning the game



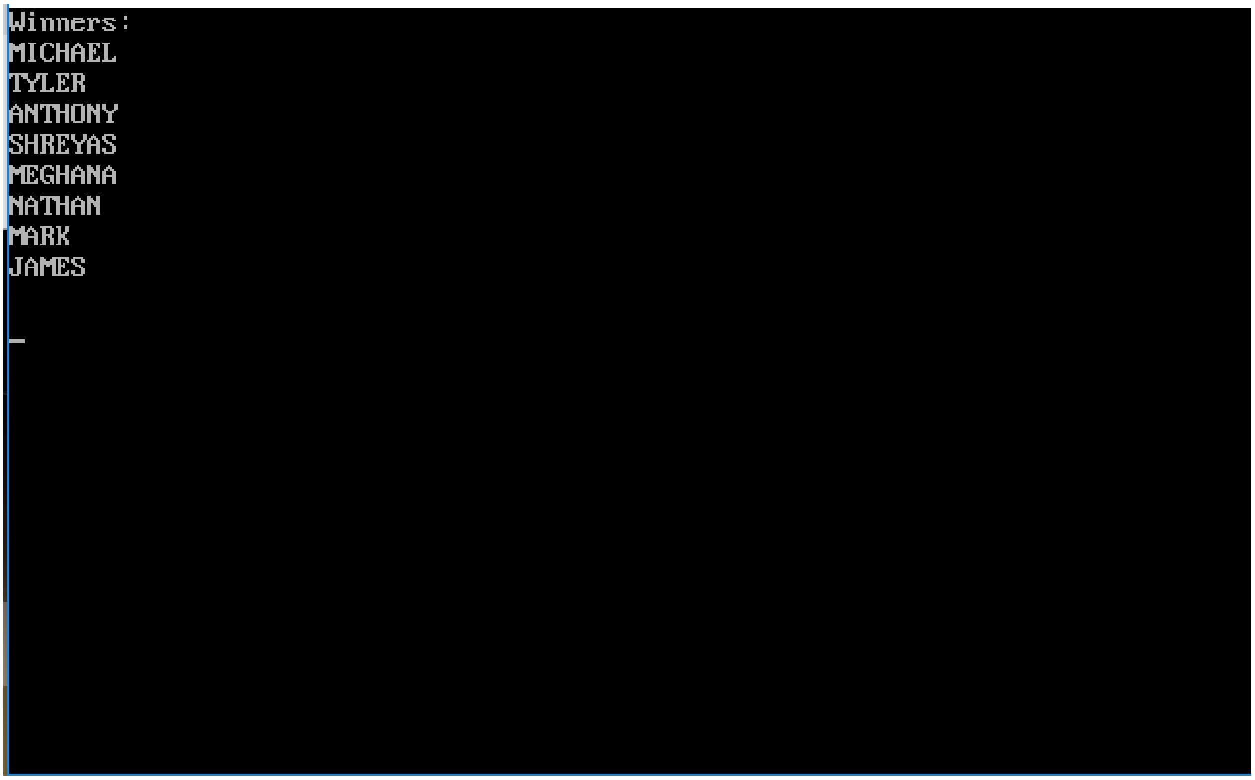
```
Congrats! JAMES you won the game.
```

Menu displayed after the game



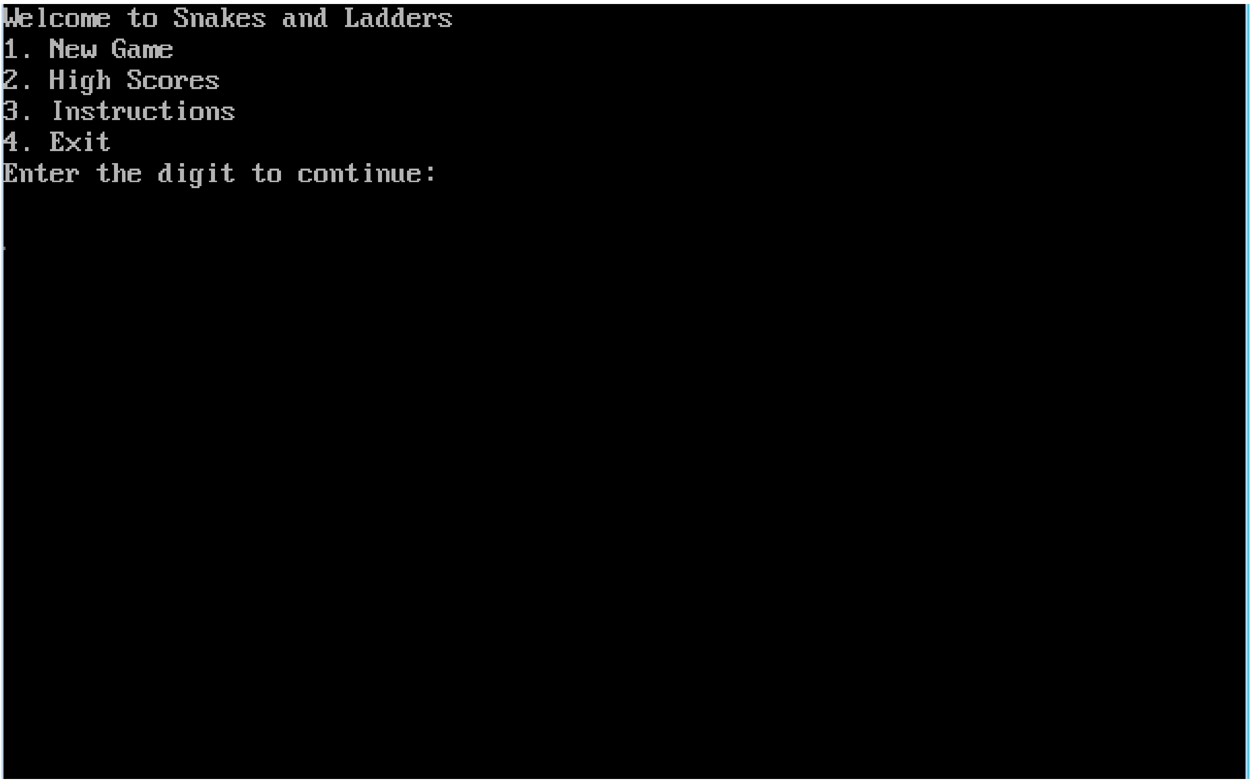
```
Welcome to Snakes and Ladders
1. New Game
2. High Scores
3. Instructions
4. Exit
Enter the digit to continue:
```

High scores displayed after the game



```
Winners:
MICHAEL
TYLER
ANTHONY
SHREYAS
MEGHANA
NATHAN
MARK
JAMES

_
```

Quitting the game



```
Welcome to Snakes and Ladders
1. New Game
2. High Scores
3. Instructions
4. Exit
Enter the digit to continue:
```

Conclusion

At the end of this project I was able to:

1. Apply graphics to my program and use graphics.h.

2. Explain object oriented programming concepts.

3. Apply them for modeling games.

4. Demonstrate the ability to develop and derive new class structures and organize them to form a game.

Bibliography

1. Grade 11 Computer Science with C++ - Sumita Arora
2. Grade 12 Computer Science with C++ - Sumita Arora
3. Google Search Engine