|  |  |  |
| --- | --- | --- |
|  | **Nitte Meenakshi Institute of Technology**  (AN AUTONOMOUS INSTITUTION AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM)  PB No. 6429, Yelahanka, Bangalore 560-064, Karnataka  Telephone: 080- 22167800, 22167860  Fax: 080 - 22167805 |  |

**COURSE PROJECT REPORT(18CS36)**

on

**SNAKE AND LADDER GAME**

*Submitted in partial fulfillment of the requirement for the award of Degree of*

*Bachelor of Engineering*

*in*

*Computer Science and Engineering*

Submitted by:

|  |  |
| --- | --- |
| Kartikeya Shorya | 1NT18CS069 |
| Nitish Raj | 1NT18CS110 |
| Harshil Dhingra | 1NT18CS055 |
| Anubhav Yadav | 1NT18CS017 |

**Department of Computer Science and Engineering**

**( NBA Accredited till 2020)**

2019-2020

|  |  |  |
| --- | --- | --- |
|  | **Nitte Meenakshi Institute of Technology**  (AN AUTONOMOUS INSTITUTION AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM)  PB No. 6429, Yelahanka, Bangalore 560-064, Karnataka  Telephone: 080- 22167800, 22167860  Fax: 080 - 22167805 |  |

**Department of Computer Science and Engineering**

**CERTIFICATE**

This is to certify that the Course Project titled “Snake And Ladder Game”is an authentic work carried out by **Kartikeya Shorya (1NT18CS069), Nitish Raj (1NT18CS110), Harshil Dhingra (1NT18CS055), Anubhav Yadav(1NT18CS017)** bonafide students of **Nitte Meenakshi Institute of Technology**, Bangalore in partial fulfilment for the award of the degree of ***Bachelor of Engineering*** in COMPUTER SCIENCE AND ENGINEERING of Visvesvaraya Technological University, Belagavi during the academic year ***2019-2020.***

|  |  |  |
| --- | --- | --- |
| **Signature of the Faculty Incharge**  **(Vinay T R)** |  | **Name and Signature of the HOD**  **Dr. Thippeswamy M.N** |

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our effort with success. we express our sincere gratitude to our Principal **Dr. H. C. Nagaraj**, Nitte Meenakshi Institute of Technology for providing facilities.

We wish to thank our HoD**, Dr. Thippeswamy M.N** for the excellent environment created to further educational growth in our college. We also thank him for the invaluable guidance provided which has helped in the creation of a better technical report.

Thanks to our Subject Faculty **Mr. Vinay T R** for guiding and helping us out in the successful completion of this presentation.

Signature

|  |  |
| --- | --- |
| **Kartikeya Shorya** | **1NT18CS069** |
| **Nitish Raj** | **1NT18CS110** |
| **Harshil Dhingra** | **1NT18CS055** |
| **Anubhav Yadav** | **1NT18CS017** |

Date:

# CONTENTS

# INTRODUCTION ---------------------------------------------------------- 5

* 1. Abstract ----------------------------------------------------------- 6

### SYSTEM STUDY ----------------------------------------------------------- 7

* 1. Existing System -------------------------------------------------- 8
  2. Gameplay ---------------------------------------------------------- 9

1. **SYSTEM SPECIFICATION** ------------------------------------------------ **10**
   1. Software Requirements --------------------------------------- 11
   2. Hardware Requirements -------------------------------------- 11
2. SYSTEM ANALYSIS ------------------------------------------------------- 12
   1. Feasibility Analysis ---------------------------------------------- 13

### Economic & Technical Feasibility --------------------------- 13

### Social & Behavioral Feasibility ------------------------------- 14

1. SYSTEM DESIGN ---------------------------------------------------------- 15
   1. Input Design ------------------------------------------------------ 16
   2. Output Design ---------------------------------------------------- 16
   3. Interface Design ------------------------------------------------- 17
   4. Procedural and Functional Design -------------------------- 18
   5. Data Flow Diagram ---------------------------------------------- 18
   6. System Flow Chart ---------------------------------------------- 20

# IMPLEMENTATION AND TESTING ----------------------------------- 22

* 1. Testing ------------------------------------------------------------- 23

# CONCLUSION -------------------------------------------------------------- 24

# APPENDIX ------------------------------------------------------------------ 26

* 1. Source Code ------------------------------------------------------ 27
  2. Screen Shots ------------------------------------------------------ 34

# BIBILIOGRAPHY & REFERENCE --------------------------------------- 37

# INTRODUCTION

**1.1 ABSTRACT**

Snake and ladder is well known game among children even among matured people.

The rules and regulation of the game are as well-known as the game. The case study meant for implementing this game without losing its interesting and attraction. The game is in only in one mode, i.e. The computer itself will act as the both players and will show the winner among them.

The user can interact with the game using either keyboard. The number and position of the ladder and snake are generated fixed.

Random mode numbering is used for this action in which system automatically feeds random inputs for the game.

### SYSTEM STUDY

2.1 Existing System

Snakes and Ladders is an ancient Indian board game regarded today as a worldwide classic. It is played between two or more players on a gameboard 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, according to die rolls, from the start (bottom square) to the finish (Top Square), helped or hindered by ladders and snakes respectively.

The historic version had root in morality lessons, where a player's progression up the board represented a life journey complicated by virtues (ladders) and vices (snakes).

There was not a good GUI for Snake & ladder game for DOS operating system. We are trying to develop a system which makes, look and feel very interesting to play the game.

Snakes and Ladders originated in India as part of a family of dice board games, that included Gyanchauper and pachisi (present-day Ludo and Parcheesi). It was known as moksha patam or vaikunthapaali or paramapadasopaanam (the ladder to salvation). The game made its way to England and was sold as "Snakes and Ladders", then the basic concept was introduced in the United States as Chutes and Ladders (an "improved new version of England's famous indoor sport") by game pioneer Milton Bradley in 1943.

**2.2 Game Play**

Each player starts with a token on the starting square (usually the "1" grid square in the bottom left corner, or simply, the imaginary space beside the "1" grid square) and takes turns to roll a single die to move the token by the number of squares indicated by the die roll.

Tokens follow a fixed route marked on the game board which usually follows a boustrophedon (ox-plow) track from the bottom to the top of the playing area, passing once through every square. If, on completion of a move, a player's token lands on the lower-numbered end of a "ladder", the player moves the token up to the ladder's higher-numbered square. If the player lands on the higher-numbered square of a "snake" (or chute), the token must be moved down to the snake's lower-numbered square.

If a player rolls a 6, the player may, after moving, immediately take another turn; otherwise play passes to the next player in turn.

The player who is first to bring their token to the last square of the track is the winner.

# SYSTEM SPECIFICATION

3.1 Software Requirements

Operating System : Windows 10

Developing software : Dev-C++

Language used : C++

3.2 HardwareRequirements

Processor : PC with a Pentium II-class.

Memory : 256 MB RAM.

Hard disk space : 2 .5 GB installation drive.

Monitor : 15 inch Monitor

Display Type : Super VGA (800x600).

**SYSTEM ANALYSIS**

4.1 Feasibility Analysis

The main aim of the feasibility study activity is to determine whether it would be financially and technically feasible to develop the product.

The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behavior of the system.

In this project there are many details available related to the Snake & ladder game and its functions.

### 4.2 Economic & Technical Feasibility

This is concerned with specifying equipment and software that will successfully satisfy the user requirement, in examining technical feasibility, configuration of the system is given more importance than the actual make of hardware.

Snake & ladder game is developed for DOS operating system and most of the latest version of Windows series OS is supported with DOS OS. Tools development for the use of the system is Turboc C compiler.

### C Programming Language

In [computing](http://en.wikipedia.org/wiki/Computing), C is a general-purpose [programming language](http://en.wikipedia.org/wiki/Programming_language) initially developed by [Dennis Ritchie](http://en.wikipedia.org/wiki/Dennis_Ritchie) between 1969 and 1973 at [AT&T Bell Labs](http://en.wikipedia.org/wiki/AT%26T_Bell_Labs).[[4]](http://en.wikipedia.org/wiki/C_(programming_language)#cite_note-ie-4) Like most [imperative languages](http://en.wikipedia.org/wiki/Imperative_language) in the [ALGOL](http://en.wikipedia.org/wiki/ALGOL) tradition, C has facilities for [structured programming](http://en.wikipedia.org/wiki/Structured_programming) and allows [lexical variable scope](http://en.wikipedia.org/wiki/Lexical_scope) and [recursion](http://en.wikipedia.org/wiki/Recursion_(computer_science)), while a [static type system](http://en.wikipedia.org/wiki/Static_type_system) prevents many unintended operations.

Its design provides constructs that map efficiently to typical [machine instructions](http://en.wikipedia.org/wiki/Machine_instruction), and therefore it has found lasting use in applications that had formerly been coded in assembly, most notably [system software](http://en.wikipedia.org/wiki/System_software) like the [Unix](http://en.wikipedia.org/wiki/Unix) computer [operating system](http://en.wikipedia.org/wiki/Operating_system).

Certain aspects of the ANSI C standard are not defined exactly by ANSI. Instead, each implementor of a C compiler is free to define these aspects individually. This chapter tells how Borland has chosen to define these implementation-specific standards. The section numbers refer to the February 1990 ANSI Standard.

### Programming Tools

**Turbo** C**++** was a [C++](http://en.wikipedia.org/wiki/C%2B%2B) compiler and [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) and computer language originally from [Borland](http://en.wikipedia.org/wiki/Borland). Most recently it was distributed by [Embarcadero Technologies](http://en.wikipedia.org/wiki/Embarcadero_Technologies).

### 4.3 Social &Behavioral Feasibility

This concerned with the requirements and analysis of the program. Interface and layout was designed in a manner that the major part of the system was allocated to the game player board and the right side of the screen was allocated to dice running and score board display details.

Small square box was allocated to each player for moving there icon on game board with the dice score received. Each player icon was moved according to the score received and the final result f the player was shown at the end when the square reached the cell with number 100.

**SYSTEM DESIGN**

5.1 Input Design

Input for the Snake & ladder program is dice running from each player and i.e. is feed be the system itself.

For player or user name entrance purpose the input design is designed with an input message, as shown in fig 5.1.



Fig: 5.1 : Input Style

5.2 Output Design

The output design was created with board design and 100 square cells. The Snake and ladder was also drawn on the bard game for moving and jumping dice icon of each player.

On completion of the game the winner’s name will be showcased along with the total number of the times he/she rolled the dice.

It also displays the total number of the times the dice was rolled, as shown in fig 5.2.

For eg:



Fig: 5.2: Output of the Game

**5.3 Interface Design:**

**Interface design** is the first step in the development phase for an engineering system. It is defined as “The process of applying various techniques and principles for the purpose of defining a process or a system in sufficient details to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used.

DOS operating system was not supported with an powerful GUI, so we tried to explore in giving a good GUI for our Snake & ladder program, which makes a good look and feel appearance for the players.

**Snake And Ladder Board**

The main window of the game was designed by keeping in mind that the interface style used should express the main idea of the functionalities of the program.

The window was created to make the outer section of game board as shown in fig 5.3.

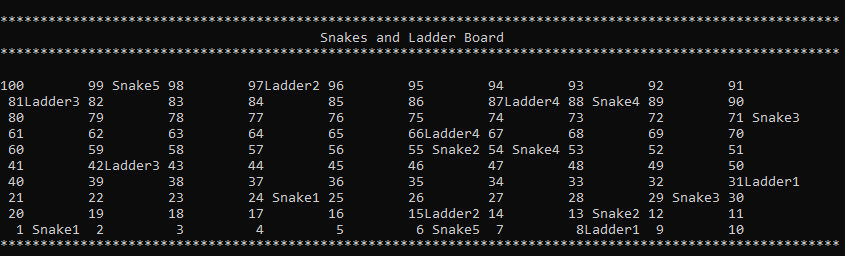


Fig: 5.3: Main Board of the Game

5.4 Procedural and Functional Design

The idea for the demonstration of the game during runtime is explained below.

The table is divided into three columns, as shown in the fig. 5.4.

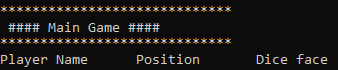


Fig: 5.4: Columns created during runtime

1. **Player Name:** This column shows the Name of the player who is rolling the dice or whose turn is there.
2. **Position:** This column shows the number of the board on which the player has reached after when the dice was rolled.
3. **Dice face:** This column display each players dice values.

**5.5 Flow Diagram**

A flow diagram is a graphical representation of the "flow" of data through an [information system](http://en.wikipedia.org/wiki/Information_system), modeling its process aspects.

Often they are a preliminary step used to create an overview of the system which can later be elaborated. Flow Diagram can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data processing](http://en.wikipedia.org/wiki/Data_processing) (structured design).We usually begin by drawing a context diagram, a simple representation of the whole system.

To elaborate further from that, we drill down to a level 1 diagram with additional information about the major functions of the system. This could continue to evolve to become a level 2 diagram.

1.

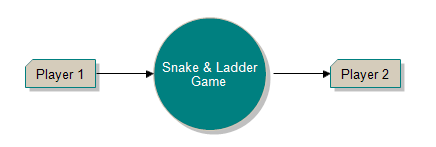


Fig 5.5: Level 1 Diagram

2.

Fig 5.6: Level 2 Diagram

**5.6 System Flow Chart**

A flowchart (also spelled flow-chart and flow chart) is a schematic representation of an algorithm or a process. A flowchart is one of the seven basic tools of quality control, which include the histogram, Pareto chart, check sheet, control chart, cause-and-effect diagram, flowchart, and scatter diagram. See Quality Management Glossary. They are commonly used in business/economic presentations to help the audience visualize the content better, or to find flaws in the process.

* Start and end symbols: Represented as circles, ovals or rounded (fillet) rectangles, usually containing the word "Start" or "End", or another phrase signaling the start or end of a process, such as "submit inquiry" or "receive product".
* Input/Output, represented as a parallelogram. Examples: Get X from the user; usually containing the word "Start" or "End", or another phrase signaling the start or end of a process, such as "submit enquiry" or "receive product".
* Arrows, showing what's called "flow of display X.
* Generic processing steps: Represented as rectangles.
* Conditional or decision, Represented as a diamond (rhombus) showing where a decision is necessary, commonly a Yes/No question or True/False test. The conditional symbol is peculiar in that it has two arrows coming out of it, usually from the bottom point and right point, one corresponding to Yes or True, and one corresponding to No or False. More than two arrows can be used, but this is normally a clear indicator that a complex decision is being taken, in which case it may need to be broken-down further.
* Arrows: Showing "flow of control". An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to.

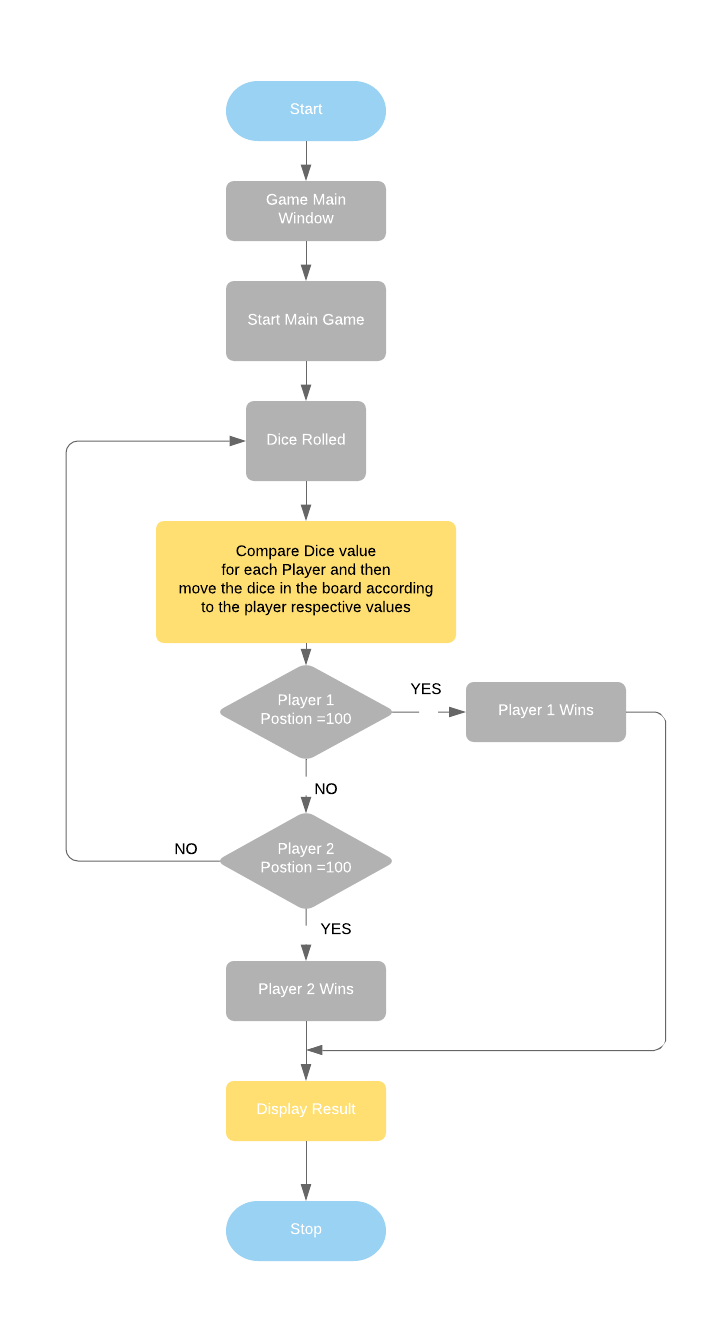


Fig: 5.7: Flow Chart of Game

**IMPLEMENTATION AND TESTING**

**6.1 Testing**

Testing is probably the most important phase for long-term support as well as for the reputation of the company. If you don’t control the quality of the software, it will not be able to compete with other products on the market.

There are multiple types of testing and these are explained in this section. Each of these has its own importance.

**Unit Testing**

Unit testing is testing one part or one component of the product. The developer usually does this when he/she has completed writing code for that part of the product. This makes sure that the component is doing what it is intended to do. This also saves a lot of time for software testers as well as developers by eliminating many cycles of software being passed back and forth between the developer and the tester. When a developer is confident that a particular part of the software is ready, he/she can write test cases to test functionality of this part of the software.

In our system **Snake & Ladder** each modules are designed and tested separately. Modules designed and tested separately are

* rollingdie() : Tested to check unbiased outcomes for dice
* Board(): Tested to draw board for playing game

**CONCLUSION**

**7.1 Conclusion**

Throughout this thesis our aim was to develop a Snake & Ladder game.

The entire game is explained and displayed in a systematic way in this report. Two players are given equal chance for winning the game.

**APPENDIX**

8.1 Source Code:

#include<iostream>

#include<cstdlib>

#include<ctime>

#include<iomanip>

using namespace std;

class Player {

public:

char name[30];

Player();

int current\_pos;

int previous\_pos;

int rollingdie();

};

Player::Player()

{

current\_pos = 1;

previous\_pos = 1;

}

int Player::rollingdie()

{

return(rand()%(6)+1);

}

class Board {

private:

int board[10][10];

public:

Board();

int nextposition(int, int);

void display();

};

Board::Board()

{

int i,j,k=1;

for(i=0; i<10;++i)

for(j=0; j<10; ++j)

board[i][j]=k++;

}

int Board::nextposition(int a, int b)

{

switch(a + b)

{

//Creating Ladders

case 8: cout<<"Great You Got Ladder at 8!!\n"<<endl;

return 31;

break;

case 15: cout<<"Great You Got Ladder at 15!!\n"<<endl;

return 97;

break;

case 42: cout<<"Great You Got Ladder at 42!!\n"<<endl;

return 81;

break;

case 66: cout<<"Great You Got Ladder at 66!!\n"<<endl;

return 87;

break;

//Creating Snakes

case 24: cout<<"Wow.... You got snake... at 24\n"<<endl;

return 1;

break;

case 55: cout<<"Wow.... You got snake... at 55\n"<<endl;

return 13;

break;

case 71: cout<<"Wow.... You got snake... at 71\n"<<endl;

return 29;

break;

case 88: cout<<"Wow.... You got snake... at 88\n"<<endl;

return 54;

break;

case 99: cout<<"Wow.... You got snake... at 99\n"<<endl;

return 6;

break;

default: return (a+b);

break;

}

}

void Board::display()

{

for(int i=9; i>=0; --i)

{

cout<<endl;

if(i%2 != 0)

{

for(int j=9; j>=0; --j)

if(board[i][j]==1 || board[i][j]==24)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake1";

else if(board[i][j]==13 || board[i][j]==55)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake2";

else if(board[i][j]==29 || board[i][j]==71)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake3";

else if(board[i][j]==54 || board[i][j]==88)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake4";

else if(board[i][j]==6 || board[i][j]==99)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake5";

else if(board[i][j]==8 || board[i][j]==31)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder1";

else if(board[i][j]==15 || board[i][j]==97)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder2";

else if(board[i][j]==42 || board[i][j]==81)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder3";

else if(board[i][j]==66 || board[i][j]==87)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder4";

else

cout<<setw(3)<<board[i][j]<<setw(7)<<" ";

}

else

for(int j=0; j<10; ++j)

if(board[i][j]==1 || board[i][j]==24)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake1";

else if(board[i][j]==13 || board[i][j]==55)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake2";

else if(board[i][j]==29 || board[i][j]==71)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake3";

else if(board[i][j]==54 || board[i][j]==88)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake4";

else if(board[i][j]==6 || board[i][j]==99)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Snake5";

else if(board[i][j]==8 || board[i][j]==31)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder1";

else if(board[i][j]==15 || board[i][j]==97)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder2";

else if(board[i][j]==42 || board[i][j]==81)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder3";

else if(board[i][j]==66 || board[i][j]==87)

cout<<setw(3)<<board[i][j]<<setw(7)<<"Ladder4";

else

cout<<setw(3)<<board[i][j]<<setw(7)<<" ";

}

}

int main()

{

srand(time(0));

Board b;

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\t\t\t\t\tSnakes and Ladder Board";

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

b.display();

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

Player player1, player2;

cout<<"Enter First player name: ";

cin>>player1.name;

cout<<"Enter Second player name: ";

cin>>player2.name;

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n #### Main Game ####\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"Player Name\t Position\tDice face\n";

int score1=0, score2=0;

int dice1\_roll=0, dice2\_roll=0;

while(1)

{

player1.previous\_pos = player1.current\_pos;

score1 = player1.rollingdie();

dice1\_roll++;

player1.current\_pos = b.nextposition(score1, player1.previous\_pos);

if(player1.current\_pos > 100)

player1.current\_pos = player1.previous\_pos;

cout<<player1.name<<"\t\t\t"<<player1.current\_pos<<"\t\t"<<score1<<endl;

if(player1.current\_pos == 100)

{

cout<<endl<<player1.name<<" won the Game "<<endl;

cout<<"The number of times "<<player1.name<<" rolled the dice: "<<dice1\_roll;

break;

}

player2.previous\_pos = player2.current\_pos;

score2 = player2.rollingdie();

dice2\_roll++;

player2.current\_pos = b.nextposition(score2, player2.previous\_pos);

if(player2.current\_pos > 100)

player2.current\_pos = player2.previous\_pos;

cout<<player2.name<<"\t\t\t"<<player2.current\_pos<<"\t\t"<<score2<<endl;

if(player2.current\_pos == 100)

{

cout<<endl<<player2.name<<" won the Game "<<endl;

cout<<"The number of times "<<player2.name<<" rolled the dice: "<<dice2\_roll;

break;

}

}

cout<<"\n\nThe total number of times dice was rolled: "<<dice1\_roll+dice2\_roll<<endl;

system("pause");

return 0;

}

8.2 Screen Shots

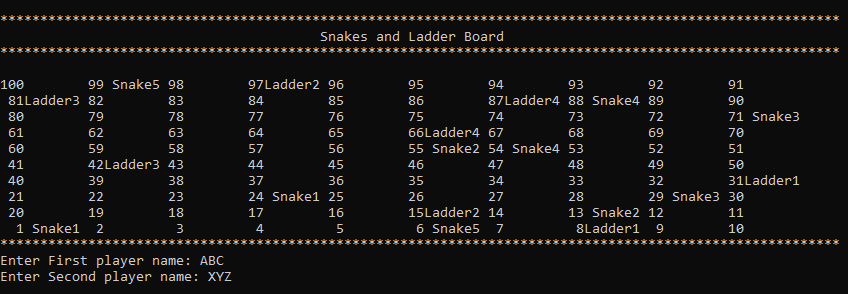


Fig: 8.1: Board & Player Name Input

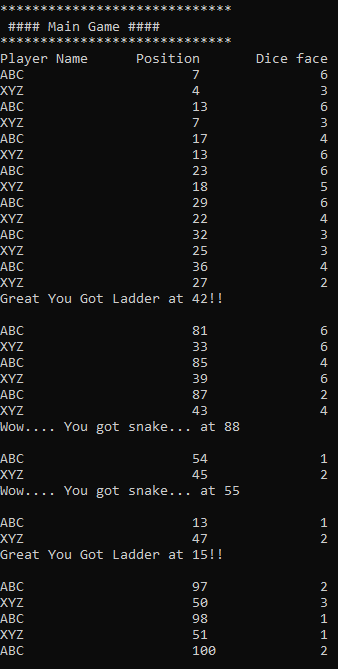


Fig: 8.2: Game Execution

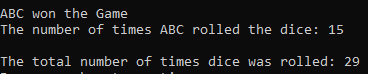


Fig: 8.3: Game Result

# BIBLIOGRAPHY

# & REFERENCE

**Bibliography:**

**Web sites:**

(1) <https://www.onlinegdb.com/online_c++_compiler>

(2) <https://www.lucidchart.com/documents#/dashboard>

**C++ Books:**

(1) Stroustrup: The C++ Programming Language,4th Edition 1979

(2) Accelerated C++: Practical Programming by Example by Andrew Koenig and Barbara E. Moo