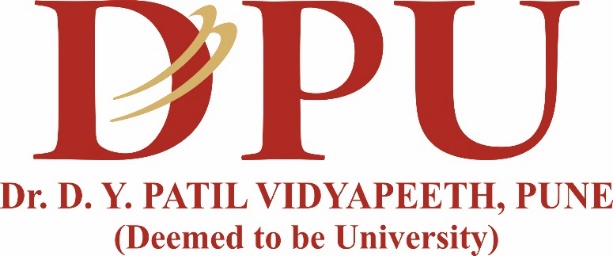
****

**DR. D. Y. PATIL VIDYAPEETH**

**PIMPRI, PUNE – 411 018**

**DR. D. Y. PATIL BIOTECHNOLOGY & BIOINFORMATICS INSTITUTE**

**TATHAWADE, PUNEs**

**A Project Based Learning Report on**

**SNAKE AND LADDER GAME**

**Submitted By**

**NAME OF MEMBERS ROLL NUMBER**

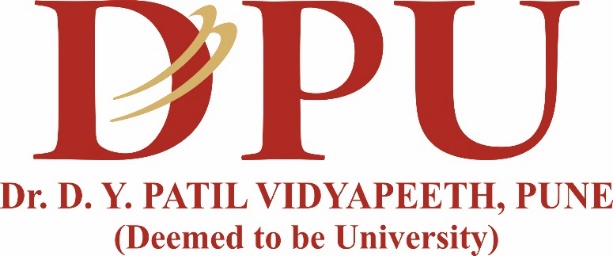
**1. ADITI BANE 05**

**2. VAISHALI KHANDAR 26**

**3. SHRAVANI JADHAV 21**

**ARTIFICIAL INTELLIGENCE & DATASCIENCE**

**ACADEMIC YEAR 2021-2022**

****

**DR. D. Y. PATIL VIDYAPEETH**

**PIMPRI, PUNE – 411 018**

**DR. D. Y. PATIL BIOTECHNOLOGY & BIOINFORMATICS INSTITUTE**

**TATHAWADE, PUNEs**

**CERTIFICATE**

**This is to certify that the Mini- Project Report entitled**

**SNAKE AND LADDER GAME**

Is a bonafide work carried out by the students under the supervision of **Mrs. Akanksha Goel** and it is submitted towards the partial fulfillment of the requirement Project Based Learning

Mrs. Akanksha Goel Prof J.K Pal

Subject TeacherDirector

**ARTIFICIAL INTELLIGENCE & DATASCIENCE**

**ACADEMIC YEAR 2021-2022**

* **ABSTRACT**

This project aims to take the pleasure and simplicity of the snake and ladder game to a new level by adding some new elements. Two players can play this game. This project is written in the C programming language. There are two players in this program. On a 10\*10 board, it is typically played. Players begin with the first tile and conclude with the 100th tile. A large number of snakes and ladders have also been erected. Snakes move a player's piece backwards while words ladders

move it forward. The game will be won by the player who reaches 100\* place first.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR.NO** | **TOPIC** | **PAGE NO** |
| 1] | PROBLEM STATEMENT | 5 |
| 2] | INTRODUCTION | 6-9 |
| 3] | ALGORITHMS | 10 |
| 4] | CODE | 11-19 |
| 5] | OUTPUTS | 20 |
| 6] | CONCLUSION | 21 |

* **PROBLEM STATEMENT**

To Implement the Snake and Ladder Game Using C programming Language

* **INTRODUCTION**

Snake and ladder is an old Indian board game that is now played all over the world. It is played on a board with numbered, gridded tiles by two or more individuals. A number of "Ladders" and "Snakes" are depicted on the board, each linking two distinct board squares. The goal of the game is to traverse between the grids of the board, from the start (Bottom Square) to the end (Top Square), based on the roll of the dice. Snake and ladder is unlike any other game. It's a fun and stress-relieving game. It arouses the user's interest. It gives endless fun.

The project is about a two-player game of snake and ladder. One participant has been chosen to begin the game. Starting from the tile, the first player rolls the dice and moves their piece by the number the die produces. After that, others repeat the process until someone throws the 100th tile or more. If a player lands at the start of a snake or a ladder, they must proceed to the finish and pass the turn to the next player.

The implementation of the code is done using following concepts:

* If:

The if statement evaluates the test expression inside parenthesis. If test expression is evaluated to True, statements inside the body of “if” is executed. If test expression is evaluated to false, statements inside the body of “if” is skipped

**if**(expression)

{

//code to be executed

}

* If else:

The if else executes the codes inside the body of if statement if test expression is true and skips the codes inside the body of else. If the test expression is false, it executes the codes inside the body of else statement and skip the codes inside the body of if.

**if**(expression)

{

//code to be executed if condition is true

}

**Else**

{

//code to be executed if condition is false

}

* If else-if ladder statement:

The if-else-if ladder statement is an extension to the if-else statement. It is used in the scenario where there are multiple cases to be performed for different conditions. In if-else-if ladder statement, if a condition is true then the statements defined in the if block will be executed, otherwise if some other condition is true then the statements defined in the else-if block will be executed, at the last if none of the condition is true then the statements defined in the else block will be executed.

**if**

{

//code to be executed if condition1 is true

}

**else** **if**

{

//code to be executed if condition2 is true

}

**else** **if**

{

//code to be executed if condition3 is true

}

**else**

{

//code to be executed if all the conditions are false

}

* Switch case:

The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possible values of a single variable called switch variable. Here, we can define various statements in the multiple cases for the different values of a single variable.

**switch**(expression)

{

**case** value1:

 //code to be executed;

**break**;

//optional

**case** value2:

 //code to be executed;

**break**;

  //optional

......

**default**:

 code to be executed **if** all cases are not matched;

}

\*The break statement is use to prevent the code running into the next case.

* While loop:

While loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given Boolean condition. It can be viewed as a repeating if statement. The while loop is mostly used in the case where the number of iterations is not known in advance.

**while**(condition)

{

//code to be executed

}

* For loop:

The **for loop in c language** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like the array and linked list.

**for** (Expression 1; Expression 2; Expression 3)

{

//code to be executed

}

\*The goto statement is known as jump statement in C. Goto is used to transfer the program control to a predefined label.

* **ALGORITHM**

STEP: 1. Start

STEP: 2. Throw the dice

STEP: 3. Move the coin

STEP: 4. Check if the coin landed on snake head

STEP: 5. If yes, slide down to the tail of the snake and then give dice to the next player to throw the dice

STEP: 6. If no, Check if the coin landed on bottom of the ladder

STEP: 7. If yes, move up the ladder and then give dice to the next player to throw the dice

STEP: 8. If no, Check if the coin has reached the last block of the game

STEP: 9. If no, give dice to the next player to throw the dice

STEP: 10. If yes, Print "You are the winner!"

STEP: 11. Stop

* **Source Code**

#include<stdio.h>

#include<stdlib.h>

int rd() //user defined function

{

int rem; //variable declaration

A:rem=rand()%7; //definition

if(rem==0) //condition

goto A;

else

return rem;

}

void displaychart(int curp,char player[4]) //user defined function

{

int i,j,t,c,sft=0,diceres,pos1,pos2; //declaration

if(curp==100) //condition

{

printf("Congratulations!!!!!! \n\nPlayer %s wins\n",player);

scanf("%\*s");

exit(0);

}

for(i=10;i>0;i--) //loop

{

t=i-1;

if((sft%2)==0)

{

c=0;

for(j=10;j>=1;j--)

{

diceres=(i\*j)+(t\*c++);

if(curp==diceres) //condition

printf("%s\t",player);

else

printf("%d\t",diceres);

}

sft++; //incrementation

}

else

{

c=9;

for(j=1;j<=10;j++) //loop

{

diceres=(i\*j)+(t\*c--);

if(curp==diceres)

printf("%s\t",player);

else

printf("%d\t",diceres);

}

sft++;

}

printf("\n\n");

}

printf("\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");

}

void main() //global function

{

int i,dice,cur\_pos1=0,cur\_pos2=0;

char ch;

while(1) //loop

{

printf("Snakes: | 25 to 9 | 65 to 40 | 99 to 1 |\nLadder: | 13 to 42 | 60 to 83 | 70 to 93 |\n\n");

printf("Choose your option\n");

printf("[1] Player 1 plays\n");

printf("[2] Player 2 plays\n");

printf("[3] Exit\n");

scanf("%s",&ch);

switch(ch) //condition

{

case '1':dice=rd();

system("cls");

printf("\n\n\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");

printf("\t\t\t Snakes And Ladders\n");

printf("\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");

cur\_pos1=dice+cur\_pos1;

if(cur\_pos1<101)

{

if(cur\_pos1==99)

{

displaychart(1,"-P1-"); //snake

}

if(cur\_pos1==65)

{

displaychart(40,"-P1-"); //snake

}

if(cur\_pos1==25)

{

displaychart(9,"-P1-"); //snake

}

if(cur\_pos1==70)

{

displaychart(93,"-P1-"); //ladder

}

if(cur\_pos1==60)

{

displaychart(83,"-P1-"); //ladder

}

if(cur\_pos1==13)

{

displaychart(42,"-P1-"); //ladder

}

Else

{

displaychart(cur\_pos1,"-P1-");

}

printf("\t\t\t\tDice = %d\n",dice);

printf("\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");

}

Else

{

cur\_pos1=cur\_pos1-dice;

printf("Range exceeded of Player 1.\n");

displaychart(cur\_pos1,"-P1-");

}

printf("Player 2 position is %d\n\n",cur\_pos2);

break;

case '2':dice=rd();

system("cls");

printf("\n\n\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");

printf("\t\t\t Snakes And Ladders\n");

printf("\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");

cur\_pos2=dice+cur\_pos2;

if(cur\_pos2<101)

{

if(cur\_pos2==99) //snake

{

displaychart(1,"$-P2-");

}

if(cur\_pos2==65) //snake

{

displaychart(40,"-P2-");

}

if(cur\_pos2==25) //snake

{

displaychart(9,"-P2-");

}

if(cur\_pos2==70) //ladder

{

displaychart(93,"-P2-");

}

if(cur\_pos2==60) //ladder

{

displaychart(83,"-P2-");

}

if(cur\_pos2==13) //ladder

{

displaychart(42,"-P2-");

}

Else

{

displaychart(cur\_pos2,"-P2-");

}

printf("\t\t\t\tDice = %d\n",dice);

printf("\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\xcd\n");

}

Else

{

cur\_pos2=cur\_pos2-dice;

printf("Range exceeded of Player 2.\n");

displaychart(cur\_pos2,"-P2-");

}

printf("Player 1 position is %d\n\n",cur\_pos1);

break;

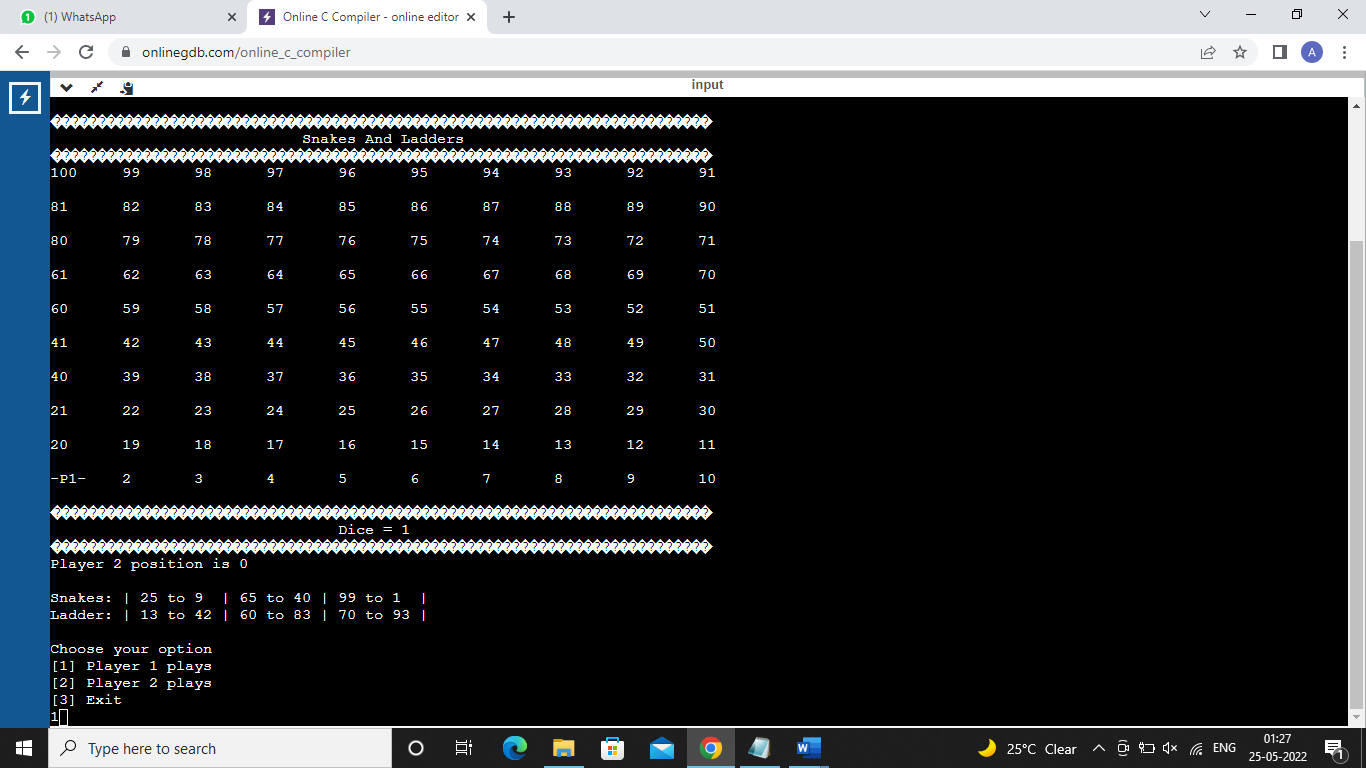
case '3':exit(0);

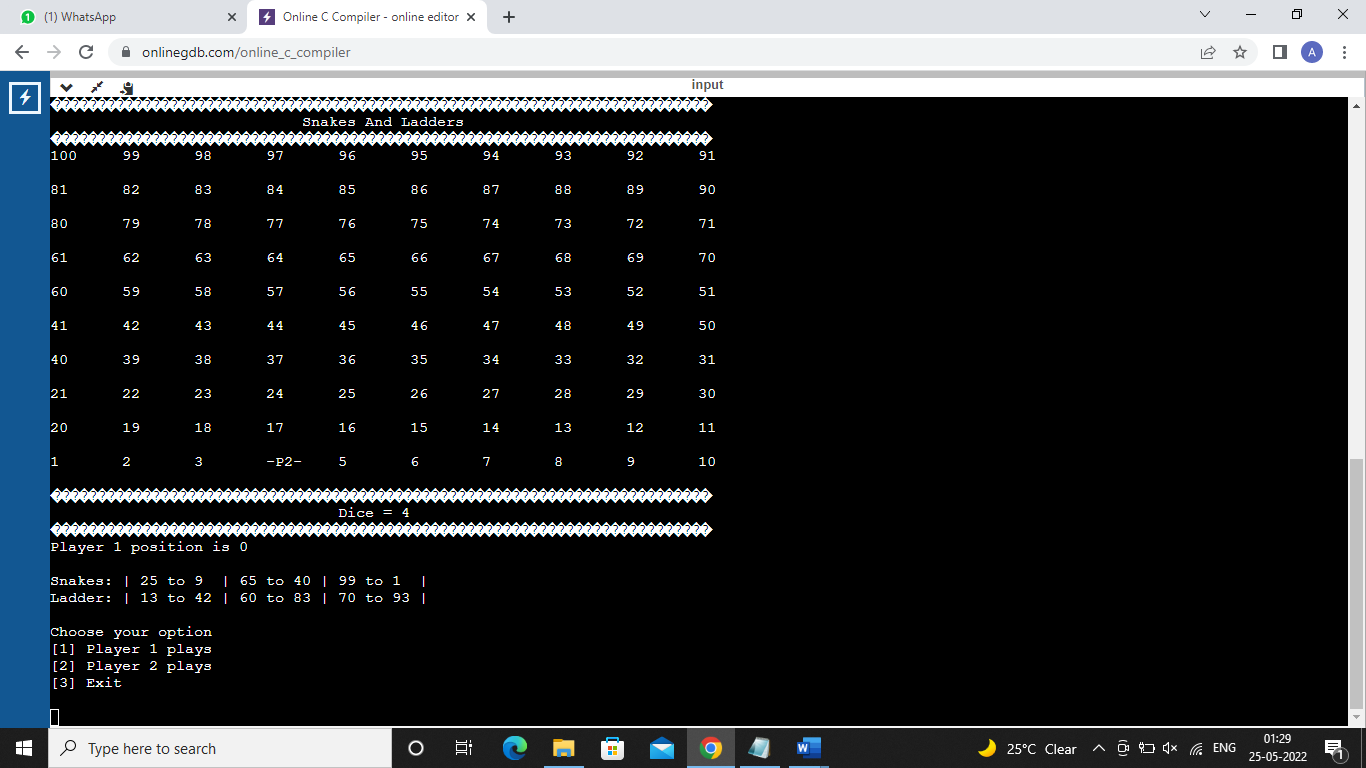
break;

default:printf("Incorrect choice.Try Again\n");}

} }

* Output





* Conclusion

The project snake and ladder game was successfully implemented using the C programming language.