A) SNAKE AND LADDER GAME

B) DECODE A PATTERN

*Design and analysis of algorithm (DAA) minor project report submitted by*

S.DEEPAK (REG. NO: 171FA04428)

V.ESWAR (REG. NO: 171FA04435)

P.SARANYU (REG. NO: 171FA04484)

*in partial fulfillment for the award of the degree of*

BACHELOR OF TECHNOLOGY

*in*

COMPUTER SCIENCE AND ENGINEERING

*under the supervision of*

Mr. RAJ KUMAR

A close up of a sign

Description automatically generated

**VFSTR UNIVERSITY**

Declared asDeemed to be University under sec.3 of the UGC Act, 1956

**Vadlamudi, Guntur - 522213, INDIA.**

**APRIL 2019**

|  |  |
| --- | --- |
| **TABLE OF CONTENTS** |  |
| Title | Page No. |
| **ABSTRACT** | 1 |
| **INTRODUCTION** | 2 |
| **SOFTWARE REQUIREMENTS** | 5 |
| **HARDWARE REQUIREMENTS** | 6 |
| **SOURCE CODE** | 7 |
| **SCREENSHOTS** | 11 |
| **CONCLUSION** | 15 |
| **REFERENCE** | 16 |
|  |  |

**SNAKE AND LADDER GAME**

**ABSTRACT**

The idea is to consider the given snake and ladder board as a directed graph with number of vertices equal to the number of cells in the board.

The problem reduces to finding the shortest path in a graph.

Every vertex of the graph has an edge to next six vertices if next 6 vertices do not have a snake or ladder.

If any of the next six vertices has a snake or ladder, then the edge from current vertex goes to the top of the ladder or tail of the snake

The hardest part here in solving the Snakes and Ladder by graphs is correctly determining what your Vertices and Edges are.

Once you get that, all you have to do is the Breadth First Search in the resultant graph. Then you can get the shortest path from Vertex 1 to Vertex 100.

Now, try getting the Adjacency Lit correct and simply call the BFS method. I’m sure you will succeed if you put in a little dedication. But for those who tried and failed.

For example, consider the board shown on right side , the minimum number of dice throws required to reach cell 30 from cell 1 is 3. Following are steps.

a) First throw two on dice to reach cell number 3 and then ladder to reach 22  
b) Then throw 6 to reach 28.  
c) Finally through 2 to reach 30.

There can be other solutions as well like (2, 2, 6), (2, 4, 4), (2, 3, 5).. etc.

Time complexity of the above solution is O(N) as every cell is added and removed only once from queue.

And a typical enqueue or dequeue operation takes O(1) time.

**INTRODUCTION**

* Snake and ladder game:-

This game is as similar to the games which found in mobiles.

As similar to mobile game we can throw the die using numeric key.

This game is having two option they are

1. user vs computer
2. user1 vs user2

Algorithm: -

Step1: game begins

Step2: roll of the die

Step3: move ahead as many steps as the number on the die

(is there is a ladder? if YES , climb it & go to step5.if NO go to step4)

Step4:is there a snake?

If YES , go down & goto step6

If NO , remain where your are & goto step5

Step5:have you reached end?

If YES , you have won the game . goto step7

If NO , goto step6

Step6:next players turn – goto step 2

Step7:game over

* In the beginning of the program, I added all the edges as though the Game Board had no snakes or ladders at all (these number of edges is what I printed), then, I removed the respective edges concerning the snakes and ladders one-by-one.
* When a Vertex ‘n’ has a ladder or a snake, we are supposed to replace the corresponding edges as I depicted in the pictures above, for that, I replaced the Vertex **n**‘s edge with the new value, in Vertices (n – 1), (n – 2), (n – 3),
* (n – 4), (n – 5), (n – 6).
* Because it is only in these vertices that you can find an edge of vertex n.
* I put all this replacing stuff in a function replace() which takes the Linked List and searches for a value ‘oldVertex’ and replaces it with the value ‘newVertex’ when it finds it.
* I used an extra element in my array to make them 1 – index based.
* The number of moves to complete the shortest path would be the level of the last vertex, Vertex 100. Why…?! Think about it for a minute and you’ll get it…!
* I have added a recursive function, printShortestPath() which recursively looks at the parent of each vertex until the start vertex is reached. It keeps printing vertices as the recursion stack keeps popping out, thus we get the path in a reverse order.

STATEMENT :

Given a snake and ladder board, find the minimum number of dice throws required to reach the destination or last cell from source or 1st cell.

OBJECTIVE :

* Over goal is to create board by using array.
* There is no strategy its based on purely Luck.
* Here we make use of array’s and random()

**DECODE A PATTERN**

**ABSTRACT**

The idea is simple, we generate all terms from 1 to n. First two terms are initialized as “1” and “11”, and all other terms are generated using previous terms. To generate a term using previous term, we scan the previous term. While scanning a term, we simply keep track of count of all consecutive characters. For sequence of same characters, we append the count followed by character to generate the next term.

To generate a member of the sequence from the previous member, read off the digits of the previous member, counting the number of digits in groups of the same digit. For example:

1 is read off as "one 1" or 11.

11 is read off as "two 1s" or 21.

21 is read off as "one 2, then one 1" or 1211.

1211 is read off as "one 1, one 2, then two 1s" or 111221.

111221 is read off as "three 1s, two 2s, then one 1" or 312211.

The look-and-say sequence was introduced and analyzed by John Conway.

**INTRODUCTION**

Decoding a pattern means finding of the nth row of that pattern

Where n is the last row of that pattern

According to the starting rows the nth row of the pattern is to be calculated and printed

Let us take an example to for decoding a pattern

Example pattern

Step1:-1  
Step2:- 11  
Step3:- 21  
Step4:- 1211  
Step5:- 111221  
Step6:- ............ (nth row)

**SOFTWARE REQUIREMENTS**

* Basic software requirement to develop program :-
  + Language: – C++.
  + Operating system – windows 7 (or) higher.
  + Compiler: - TDM GCC
  + Editor: - DEV C++

**HARDWARE REQUIREMENTS**

* Basic hardware requirements to develop program: -
  + Processor – Intel .
  + Hard disk(ROM) – 50GB (or) higher .
  + Base memory(RAM) – 2GB (or) higher.

**SOURCE CODE**

SNAKE AND LADDER GAME: -

#include<iostream>

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<time.h>

using namespace std;

int main()

{

cout<<" |SNAKE AND LADER GAME|\n";

cout<<"\n|SNAKE AND LADER BOARD|:-\n";

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

string s[10][10]={{"| 100 (L6)99 \*(S7)98 97 (L4)96 95 94 (L3)93 \*(S6)92 (L2)91 |"},{"| 81 (L1)82 \*(S5)83 84 85 86 (L5)87 88 89 90 |"},{"| +(L6)80 79 78 77 76 75 74 73 72 71 |"},{"| 61 +(L4)62 63 64 65 +(L5)66 67 68 \*(S4)69 70 |"},{"| 60 59 58 57 56 \*(S3)55 +(L3)54 53 52 (S6)51 |"},{"| 41 42 43 \*(S1)44 45 46 47 \*(S2)48 49 +(L2)50 |"},{"| 40 39 38 37 36 35 34 (S4)33 32 31 |"},{"| +(L1)21 22 23 24 25 26 27 (S7)28 29 30 |"},{"| 20 (S1&S5)19 18 17 16 15 14 13 12 11 |"},{"| 1 2 3 4 5 6 (S3)7 8 (S2)9 10 |"}};

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

{

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

{

cout<<s[i][j]<<"\t";

}

cout<<endl;

}

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

cout<<"\n(L) IN SNAKE AND LADER BOARD REPRESENTS STARTING POINT OF THE LADDER AND (L) REPRESENTS ENDING POINT OF THE LADDER\n";

cout<<"\n+(S) IN SNAKE AND LADER BOARD REPRESENTS HEAD OF THE SNAKE AND (S) REPRESENTS TAIL OF THE SNAKE\n";

int a,b,e,n;

cout<<"\nEnter single or double players \n";

cin>>n;

if(n==1)

{

a=0;

b=0;

int da=0,db=0,c=0;

int r;

while(a!=100||b!=100)

{

cout<<"press any numeric for die Rotation\n";

cin>>r;

cout<<"\n";

srand ( time(NULL) );

da=rand()%7;

if(da==0)

da=1;

cout<<da<<" is your score\n";

a=a+da;

if(a>100)

a=a-da;

if(a==100)

break;

else if(a==21)

{

a=82;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*You got Ladder\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==50)

{

a=91;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*You got Ladder\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==54)

{

a=93;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*You got Ladder\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

} else if(a==62)

{

a=96;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*You got Ladder\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

} else if(a==66)

{

a=87;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*You got Ladder\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==80)

{

a=99;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*You got Ladder\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==44)

{

a=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\* You have a Snake\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==48)

{

a=9;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*You have a Snake\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==55)

{

a=7;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==69)

{

a=33;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==83)

{

a=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==92)

{

a=51;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

else if(a==98)

{

a=28;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"You are at "<<a<<" now\n";

}

cout<<"\n";

cout<<" Your Position \n";

cout<<" "<<a<<" \n";

cout<<"\n";

db=rand()%7;

if(db==0)

db=1;

b=b+db;

cout<<"\n";

cout<<db<<" is Computer score\n";

if(b>100)

b=b-db;

if (b==100)

break;

else if(b==21)

{

b=82;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Ladder\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==50)

{

b=91;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Ladder\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==54)

{

b=93;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Ladder\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==62)

{

b=96;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Ladder\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==66)

{

b=87;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Ladder\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==80)

{

b=90;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Ladder\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==44)

{

b=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==48)

{

b=9;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==55)

{

b=7;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}else if(b==69)

{

b=33;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==83)

{

b=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==92)

{

b=51;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

else if(b==98)

{

b=28;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Snake\*\*\*\*\n";

cout<<"Computer is at "<<b<<" now\n";

}

cout<<"\n";

cout<<" Computer Position \n";

cout<<" "<<b<<" \n";

cout<<"\n";

c++;

}

if(a==100)

cout<<"\*\*\*\*\*\*Congratulations\*\*\*\*\*\n"<<"\*\*You Won in "<<++c<<" Steps\*\*";

else

cout<<"OOPS:("<<" You Lost\n"<<"Computer won in "<<++c<<" Steps";

return 0;

}

else

{

a=0;

b=0;

int da=0,db=0,c=0;

int r;

while(a!=100||b!=100)

{

cout<<"Player 1 : press any numeric for die Rotation\n";

cin>>r;

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

srand ( time(NULL) );

da=rand()%7;

if(da==0)

da=1;

cout<<da<<" is Player 1's score\n";

a=a+da;

if(a>100)

a=a-da;

if(a==100)

break;

else if(a==21)

{

a=82;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 1 got Ladder\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==50)

{

a=91;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 1 got Ladder\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==54)

{

a=93;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 1 got Ladder\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

} else if(a==62)

{

a=96;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 1 got Ladder\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

} else if(a==66)

{

a=87;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 1 got Ladder\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==80)

{

a=99;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 1 got Ladder\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==44)

{

a=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\* Player 1 have a Snake\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==48)

{

a=9;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 1 have a Snake\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==55)

{

a=7;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 1 have a Snake\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==69)

{

a=33;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 1 have a Snake\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==83)

{

a=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 1 have a Snake\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==92)

{

a=51;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 1 have a Snake\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

else if(a==98)

{

a=28;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 1 have a Snake\*\*\*\*\n";

cout<<"Player 1 is at "<<a<<" now\n";

}

cout<<"\n";

cout<<" Player 1 Position \n";

cout<<" "<<a<<" \n";

cout<<"\n";

int s,dc=0;

cout<<"Player 2 : press any key for die Rotation\n";

cin>>s;

cout<<"\n";

dc=rand()%7;

if(dc==0)

dc=1;

e=e+dc;

cout<<"\n";

cout<<dc<<" is Player 2's score\n";

if(e>100)

e=e-dc;

if (e==100)

break;

else if(e==21)

{

e=82;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 2 got a Ladder\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==50)

{

e=91;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 2 got a Ladder\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==54)

{

e=93;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 2 got a Ladder\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==62)

{

e=96;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 2 got a Ladder\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==66)

{

e=87;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 2 got a Ladder\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==80)

{

e=90;

cout<<"\*\*\*Hurray\*\*\*\n\*\*\*\*Player 2 got a Ladder\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==44)

{

e=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 2 got Snake\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==48)

{

e=9;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 2 got Snake\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==55)

{

e=7;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 2 got Snake\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}else if(e==69)

{

e=e-(69-33);

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 2 got Snake\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==83)

{

e=19;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 2 got Snake\*\*\*\*\n";

cout<<"player 2 is at "<<e<<" now\n";

}

else if(e==92)

{

e=51;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 2 got Snake\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

else if(e==98)

{

e=28;

cout<<"\*\*\*oops!\*\*\*\n\*\*\*\*Player 2 got Snake\*\*\*\*\n";

cout<<"Player 2 is at "<<e<<" now\n";

}

cout<<"\n";

cout<<" Player 2 position is at \n";

cout<<" "<<e<<" \n";

cout<<"\n";

c++;

}

if(a==100)

cout<<"\*\*\*\*\*\*Congratulations\*\*\*\*\*\n"<<"\*\*player 1 won in "<<++c<<" Steps\*\*";

else if(e==100)

{

cout<<"\*\*\*\*\*\*Congratulations\*\*\*\*\*\n"<<"\*\*player 2 won in "<<++c<<" Steps\*\*";

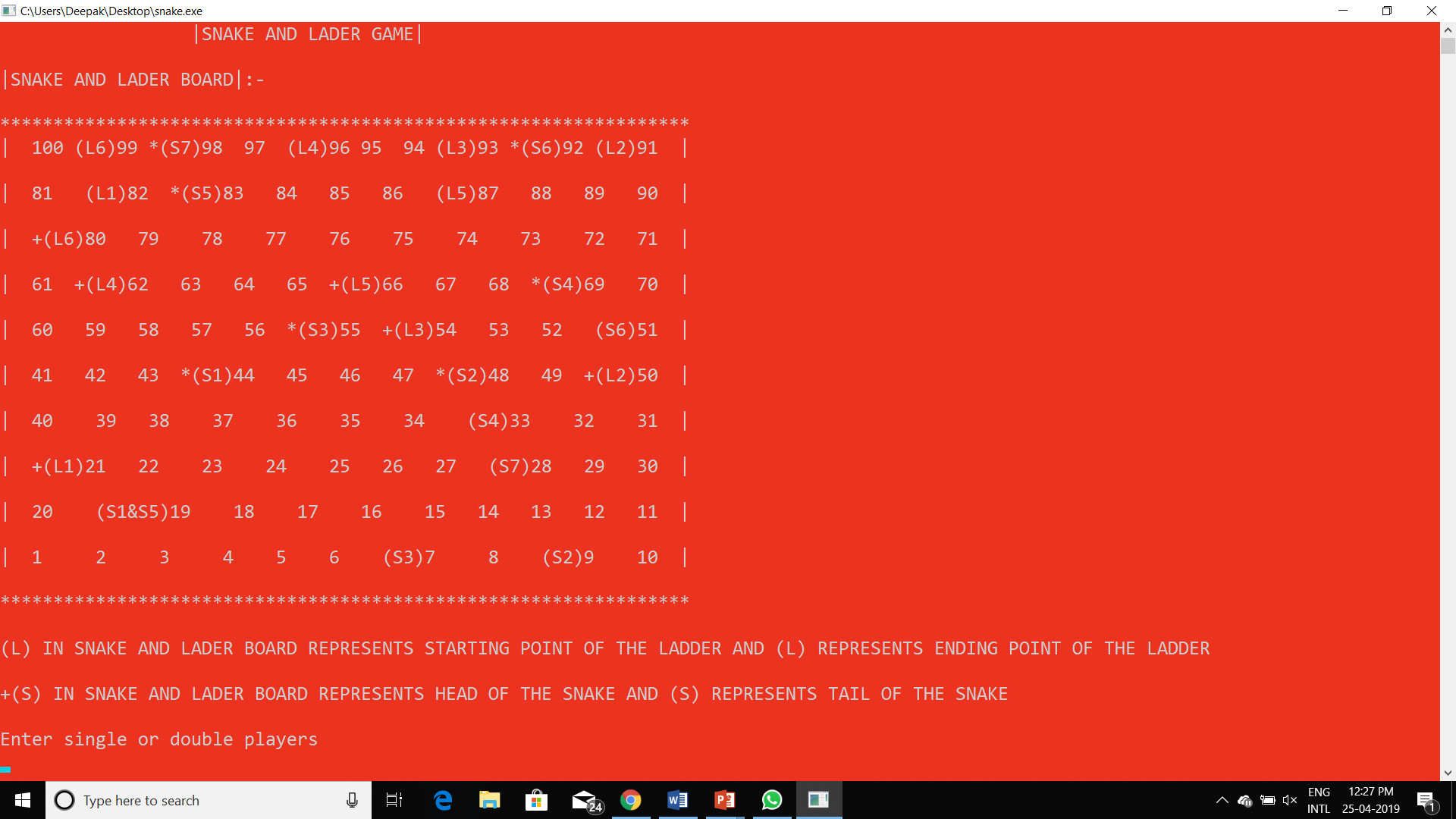
}

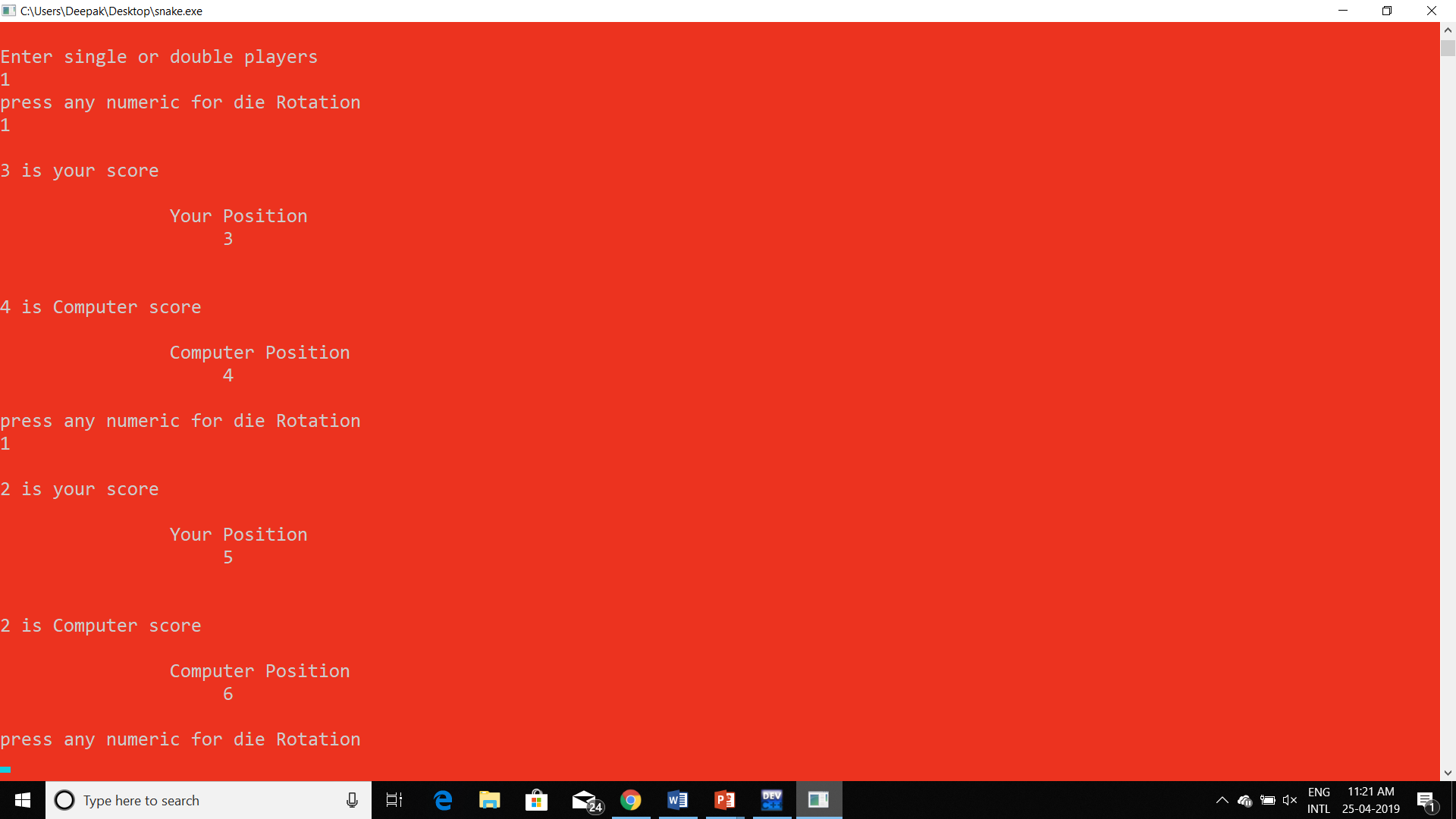
return 0;

}

}

**SCREEN SHOTS**

****

****

**SOURSE CODE**

**DECODE A PATTERN: -**

**#include<iostream>**

**#include<string>**

**using namespace std;**

**int main() {**

**int n;**

**cin >> n;**

**if(n==1)cout << 1 ;**

**else if(n==2)cout << 11;**

**else{**

**string s ="11";**

**for(int j=3;j<=n;j++)**

**{**

**int count = 1;**

**string s1="";**

**for(int i=1;i<s.length()+1;i++){**

**if(s[i]!=s[i-1]){**

**s1 = s1 + (char)(count+48);**

**s1 = s1 + s[i-1];**

**count = 1;**

**}else**

**count++;**

**}**

**s = s1;**

**}**

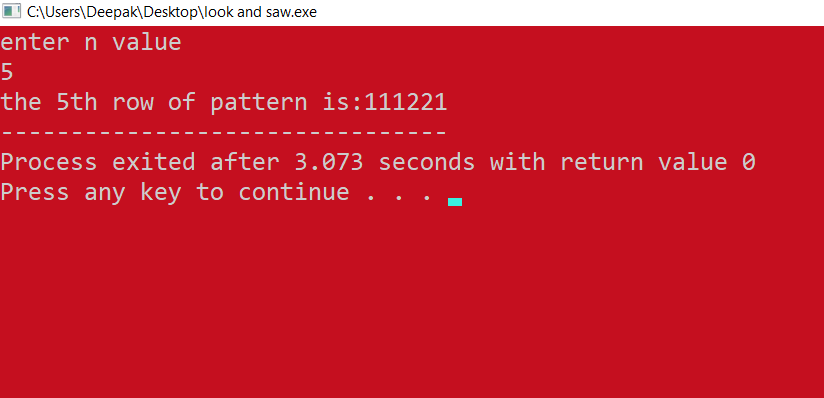
**cout << s ;**

**}**

**return 0;**

**}**

**SCREEN SHOTS**

****

**Conclusion**

1. So, you have seen the various aspects, rules and output of snake and ladder board game , you feel happy by playing this game.
2. By decode a pattern program we can get any row of that pattern.

**REFERENCES**

1. Internet, random functions, arrays
2. Look and say problem