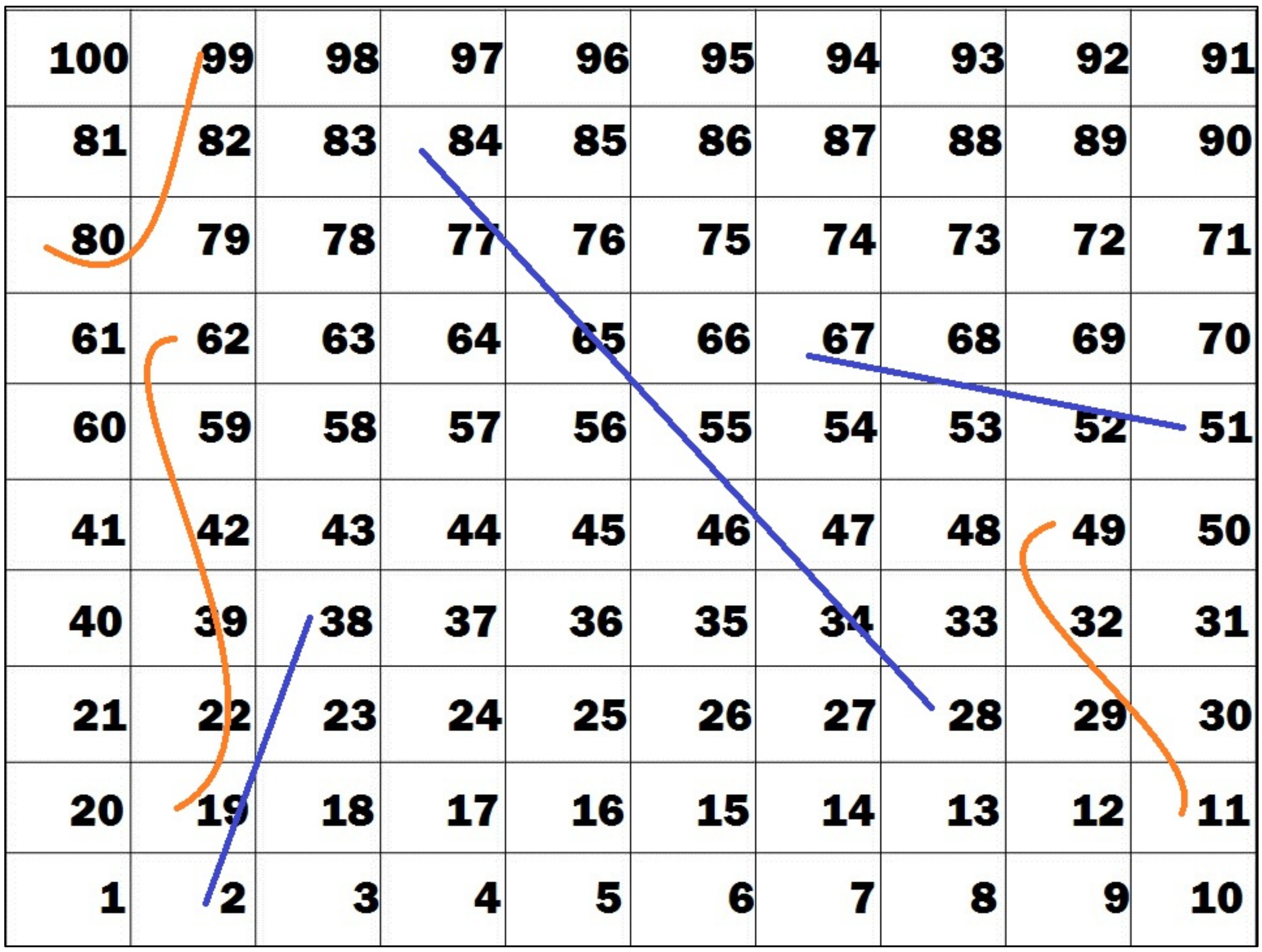


Computer Science Project

SNAKES AND LADDERS

Mohit Doshi | 12A

ACKNOWLEDGEMENTS

I would like to thank Deepa ma'am for giving me the opportunity to work on this project. I would like to thank my teacher Mrs. Smitha for her patience, support, and aid throughout the course of this project. The nature of this project required me to stay in the laboratory for extended hours and I would like to thank the students of grades 11A, 11B and 11C for freeing a workstation whenever I needed one. I would also like to thank Mr.Saffiq of Ninestars Information Technologies Pvt. Ltd. and my classmates Manav and Krishna for clearing my doubts related to the graphics portion of this project.

TABLE OF CONTENTS

SYSTEM REQUIREMENTS

1. Operating system of Windows XP or higher.
2. Dev C++ version 4.8 or higher with the graphics.h library included.
3. At least 64 megabytes (MB) of RAM.
4. At least 1.5 gigabytes (GB) of available space on the hard disk.
5. Pentium 233-megahertz (MHz) processor or faster.

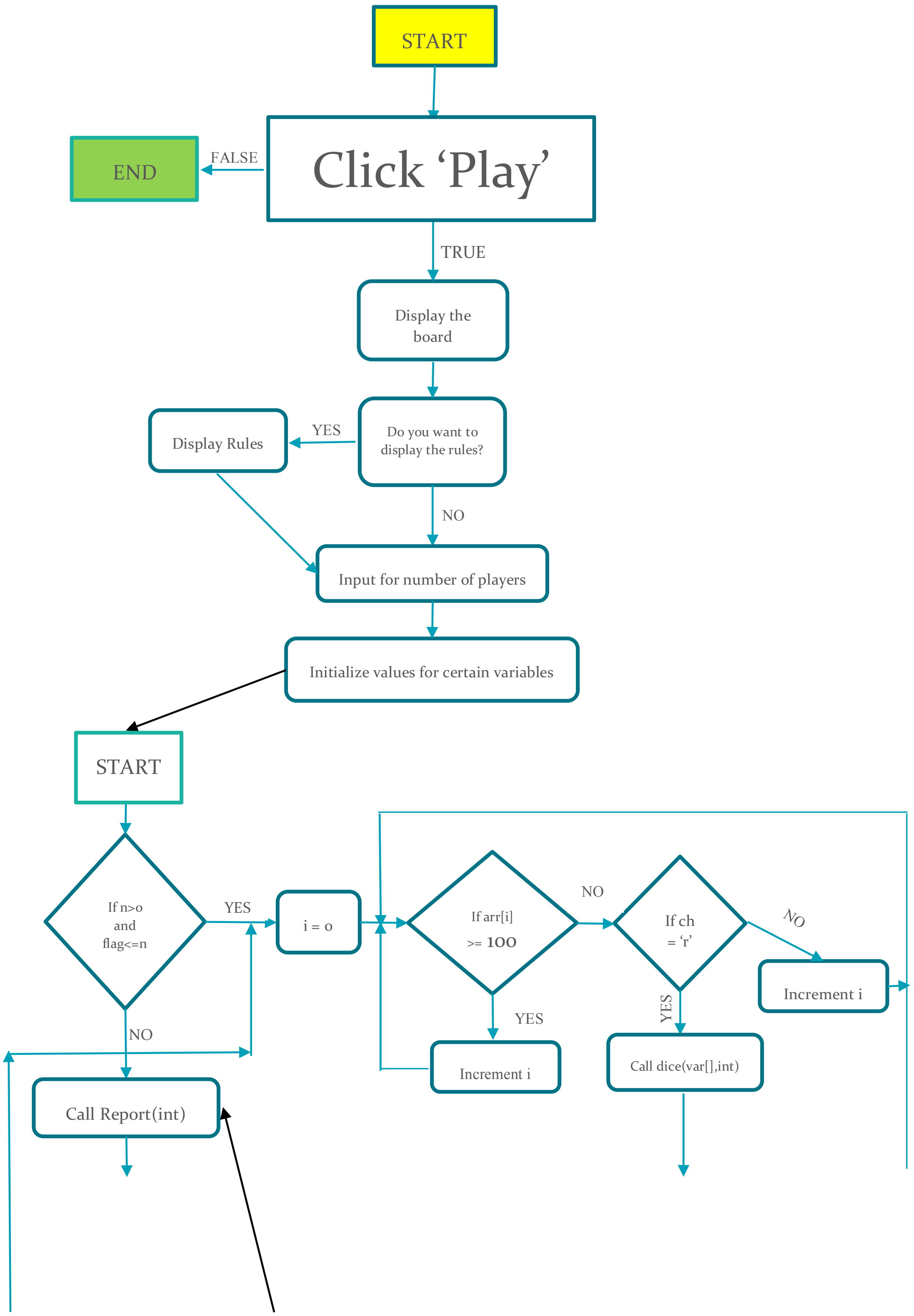
AIM OF THE PROJECT

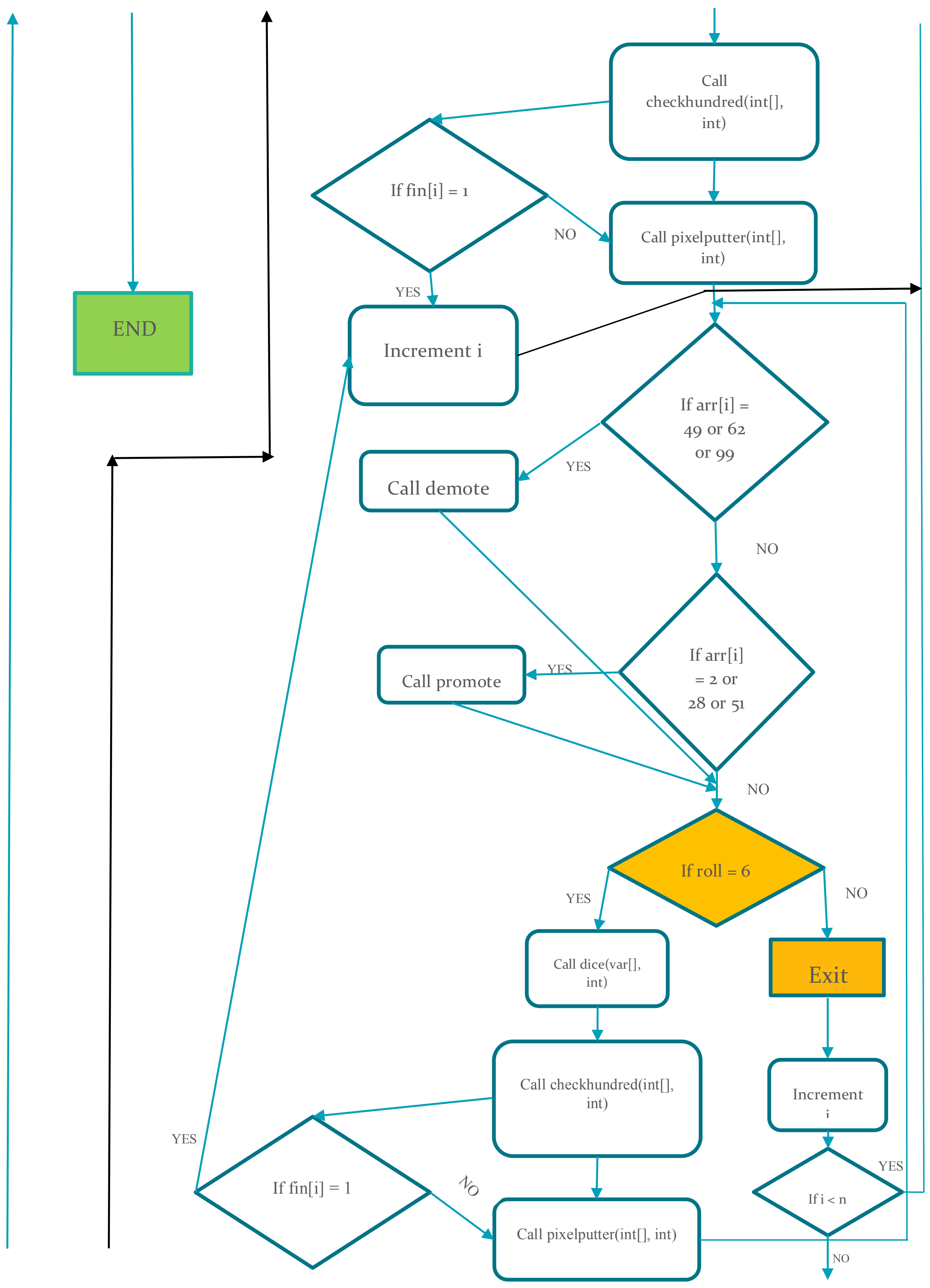
Snakes and Ladders is an ancient Indian board game. Although, regarded as a classic it is fast losing relevance in today's ever-changing world. Playing the game is known to boost counting and addition skills in children. I have always had fond memories of this board game and often played it when I was younger, with friends and family.

The aim of this project is to revive and revitalize interest in the classical board game, "Snakes and Ladders". The target audience is broad and covers people of all ages. It simulates a game of Snakes and Ladders between two people, and provides a score report in the end detailing the total number of turns, snakes and ladders encountered. Advances in consumer technology have led to the demise of many board games but an electronical format of this favoured classic will ensure that it catches up with any new advents in technology. This project can be applied to mobile, desktop, even virtual reality applications. This can ensure a broader reach and greater saturation amongst masses.

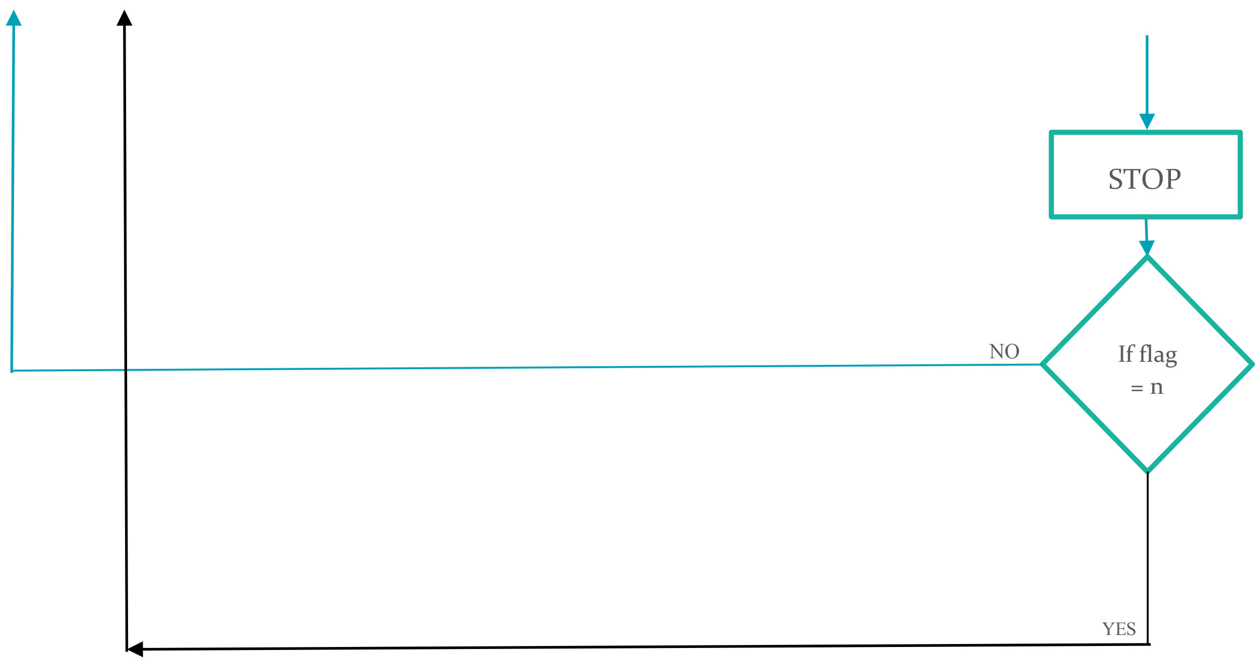
This project showcases a modernized version of a cultural and heritage symbol of India and it has the potential to regain the game's former glory in the hearts of people all around the world.

PROJECT FLOW DIAGRAM

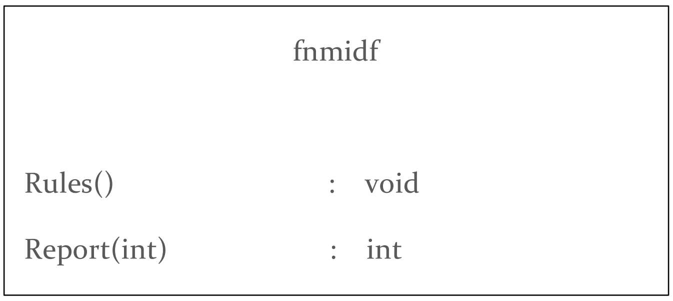


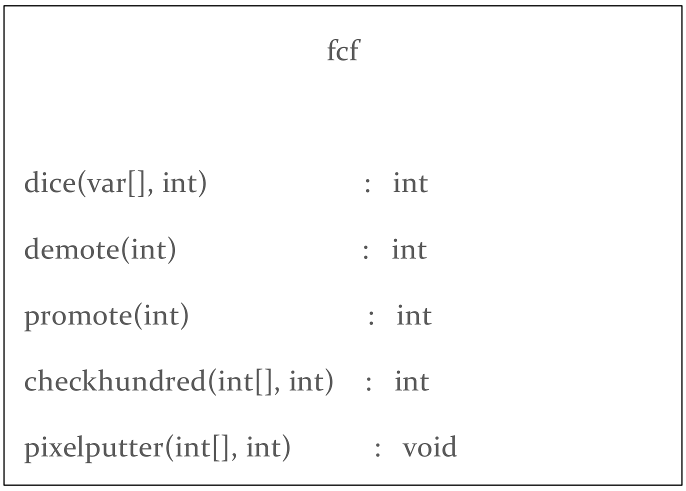


END
Call checkhundred(int[], int)
If fin[i] = 1
NO
Call pixelputter(int[], int)
YES
Increment i
If arr[i] = 49 or 62 or 99
YES
Call demote
NO
If arr[i] = 2 or 28 or 51
YES
Call promote
NO
If roll = 6
YES
NO
Call dice(var[], int)
Exit
Call checkhundred(int[], int)
Increment i
YES
If fin[i] = 1
NO
YES
If i < n
Call pixelputter(int[], int)
NO


STOP
If flag = n
NO
YES

CLASS DIAGRAM




fcf
dice(var[], int) : int
demote(int) : int
promote(int) : int
checkhundred(int[], int) : int
pixelputter(int[], int) : void

FUNCTIONS

1. **void Rules();**
   A public class of fnmidf. It reads and displays the game's rules from a data file "GamesRules.dat".

2. **int Report(int);**
   A public class of fnmidf. It reads and displays the player's report from a data file "PlayerRep.dat". It also calculates and displays the final score using the formula – (1000/number of turns) + number of ladders encountered – number of snakes encountered. It returns 0 if the game is a tie, and 1 or 2 depending on the player that won.

3. **int dice(var[], int);**
   A public class of fcf. It returns a random integer value between 1 and 6, simulating a dice roll.

4. **int demote(int);**
   A public class of fcf. It performs the function of the snake in the game by demoting the player's position. It returns the new position as an integer value.

5. **int promote(int);**
   A public class of fcf. It performs the function of the ladder in the game by promoting the player's position. It returns the new position as an integer value.

6. **int checkhundred(int[], int);**
   A public class of fcf. It returns 1 if the position of the player is greater than equal to 100 while setting the player's position to 100, and it returns 0 if the position is less than 100.

7. **void pixelputter(int[], int);**
   A public class of fcf. It fills the screen with the board and based on the player's position and player number, creates a red circle indicating Player 1's current position and a blue circle indicating Player 2's current position.

8. **int confclick();**
   A global function. It returns 1 if the user clicks the play button, hence starting the game.

9. **void readimagefile (const char* filename=NULL, int left=0, int top=0, int right=INT_MAX, int bottom=INT_MAX);**

Based on the coordinates and the file name entered, it fills the contents of the file into a required size of the graphics window.

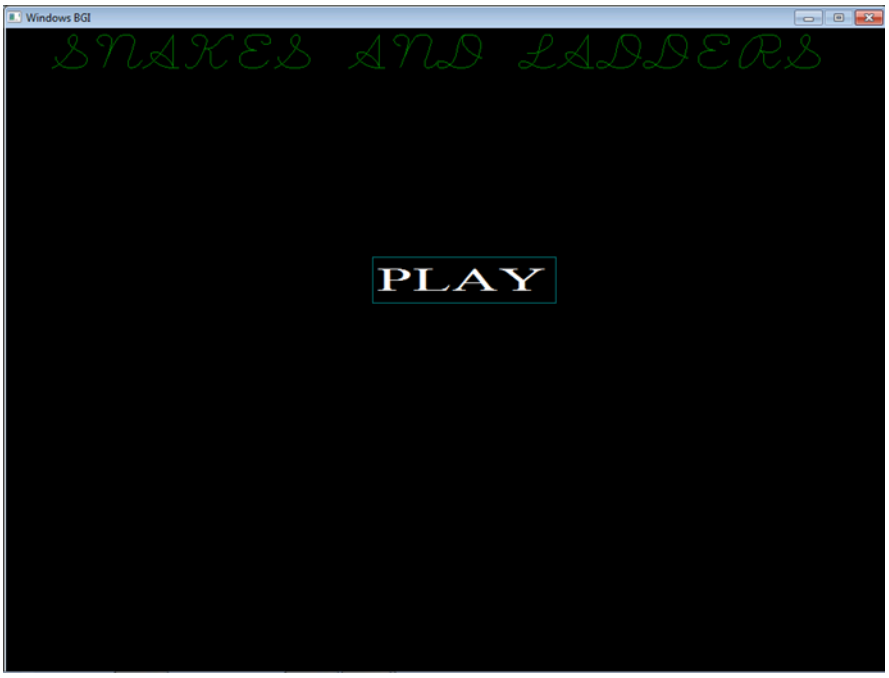
10. **void circle (int x, int y, int radius);**
x and y represent the coordinates of the centre of the circle that is plotted. The circle's size can be varied based on the value radius holds.

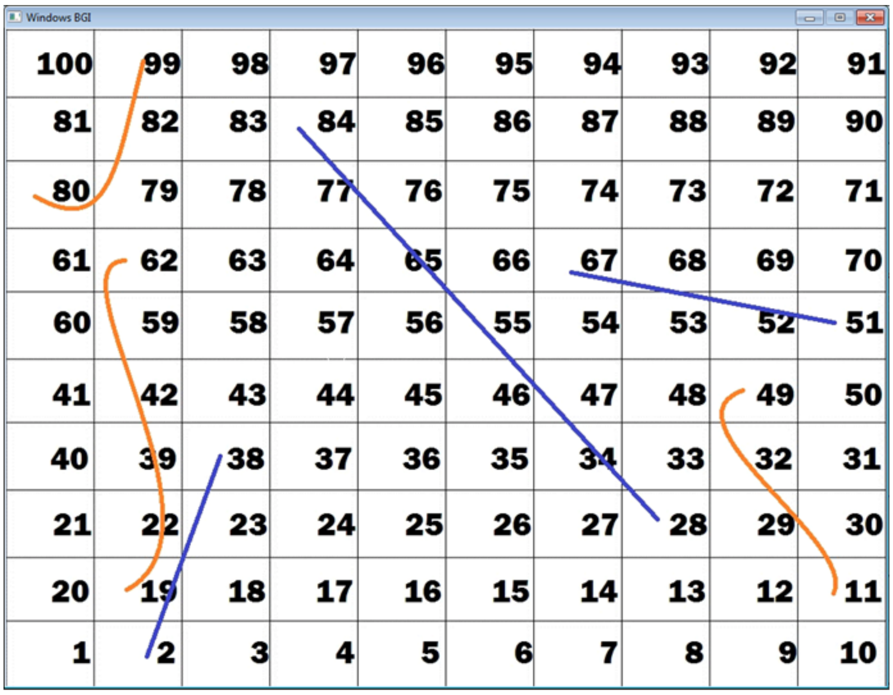
SOURCE CODE

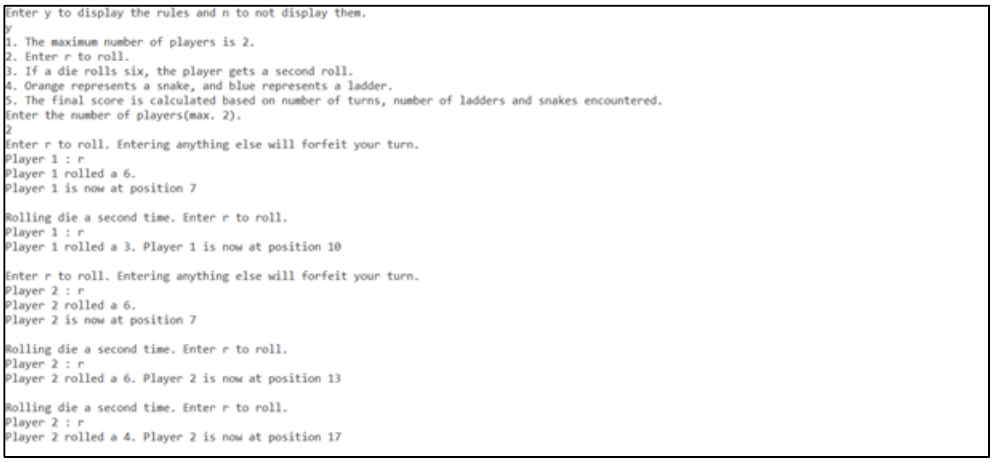
```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<iostream>
#include<time.h>
#include<stdlib.h>
#include<string.h>
#include<fstream>
using namespace std;

struct var
{
 int nt, ns, nl;
};
int prx[100] =
{18,114,210,306,402,498,594,690,786,882,882,786,690,594,498,402,306,210,114,18,18,114,21
0,306,402,498,594,690,786,882,882,786,690,594,498,402,306,210,114,18,18,114,210,306,402,
498,594,690,786,882,882,786,690,594,498,402,306,210,114,18,18,114,210,306,402,498,594,69
0,786,882,882,786,690,594,498,402,306,210,114,18,18,114,210,306,402,498,594,690,786,882,
882,786,690,594,498,402,306,210,114,18};

int pry[100] =
{716,716,716,716,716,716,716,716,716,716,639,639,639,639,639,639,639,639,639,639,562,562
,562,562,562,562,562,562,562,562,485,485,485,485,485,485,485,485,485,485,408,408,408,408
,408,408,408,408,408,408,331,331,331,331,331,331,331,331,331,331,254,254,254,254,254,254
,254,254,254,254,177,177,177,177,177,177,177,177,177,177,100,100,100,100,100,100,100,100
,100,100,23,23,23,23,23,23,23,23,23,23};

int snakedemote[3][3] = {{49,11},{62,19},{99,80}}, ladderpromote[3][3] =
{{2,38},{28,84},{51,67}};//global declarations.

class fnmidf
{
public:
       void Rules()
    {
    fstream robj;
       robj.open("GameRules.dat", ios::in|ios::binary);
       char arr[5][100];
       for(int i = 0; i < 5;i++)
       {
        robj.read((char*)&arr[i], sizeof(arr[i]));
       }
       for(int i = 0; i < 5;i++)
        cout<<arr[i]<<endl;

       robj.close();
    }
    int Report(int p)
    {
     double fsp[p];
     var ro[p];
     fstream objrep;
```

```
      objrep.open("PlayerRep.dat", ios::in|ios::binary);
      cout<<"Player reports:-\n\n";
      for(int i = 0;i < p;i++)
      {
       objrep.read((char*)&ro[i], sizeof(ro[i]));
       cout<<"Player "<<(i+1)<<endl;
       cout<<"Total number of turns/moves : "<<ro[i].nt<<endl;
       cout<<"Total number of snakes encountered : "<<ro[i].ns<<endl;
       cout<<"Total number of ladders encountered : "<<ro[i].nl<<endl;
       fsp[i] = (1000/ro[i].nt) + ro[i].nl - ro[i].ns;
          cout<<"Final score of Player "<<(i+1)<<" : "<<(int)fsp[i]<<endl<<endl;
      }
      objrep.close();
      if(p == 1)
       return 1;
      else
       {
        if(fsp[0] == fsp[1])
         return 0;
        else
        if(fsp[0]>fsp[1])
         return 1;
        else
         return 2;
          }
     }
}dob;

class fcf
{
public:
        int dice(var sobj[], int ss)
            {
            srand(time(NULL));
            sobj[ss].nt += 1;
            return(rand()%6 + 1);//returns a value between 1 and 6.
            }

        int demote(int check)
            {
             for(int i = 0;i < 3;i++)
             {
              if(check == snakedemote[i][0])
               return snakedemote[i][1];
             }
            }

        int promote(int check)
            {
             for(int i =0;i < 3;i++)
             {
              if(check == ladderpromote[i][0])
               return ladderpromote[i][1];
             }
            }

        int checkhundred(int chk[], int pos)
            {
              if(chk[pos] >= 100)
              {
                   chk[pos] = 100;
                   return 1;
              }
             else
                 return 0;
           }

       void pixelputter(int ap[], int i)
             {
              readimagefile("1to100-2.jpg",0,0,getmaxx(),getmaxy());
              //fills the entire screen with the image
              for(int c = 0; c <= (i-1); c++)
              {

              if(c == 1)//only if there are two players
               {
                setcolor(BLUE);
                if(ap[c] >= 100)
                 circle(prx[100-1]-10,pry[100-1],5);
                else
                 circle((prx[ap[c]-1]-10),(pry[ap[c]-1]), 5);
                break;
               }
               setcolor(RED);
               if(ap[c] >= 100)
                circle(prx[100-1],pry[100-1],5);
               else
               circle(prx[ap[c]-1], pry[ap[c]-1], 5);
              }
             }
}cmm;


int confclick()
    {
    int a,b;
    while(1)
    {
    if(ismouseclick(WM_LBUTTONDOWN))
    {
     getmouseclick(WM_LBUTTONDOWN,a,b);
     clearmouseclick(WM_LBUTTONDOWN);
     if((a>=400&&a<=600)&&(b>=250&&b<= 300))
         {
         return 1;
         }
     else
          return 0;
    }
    }//end of while
    }//end of confclick()



int main()
{
        initwindow(960,720);
        int x=getmaxx(),y=getmaxy();//x = 959, y = 719
        setcolor(GREEN);
        settextstyle(5,0,7);
        outtextxy(50,0,"SNAKES AND LADDERS");
        setcolor(CYAN);
        rectangle(400, 250, 600, 300);
        settextstyle(1,0,5);
        setcolor(WHITE);
        outtextxy(405,255, "PLAY");
if(confclick())//super if
{
  cleardevice();
  readimagefile("1to100-2.jpg",0,0,getmaxx(),getmaxy());
  //fills the entire screen with the image
  circle(360,360,10);
  char ch;
  int n, roll, count = 0, flag = 0, wr= 0;

        cout<<"Enter y to display the rules and n to not display them.\n";
        cin>>ch;
        if(ch == 'y' || ch == 'Y')
         dob.Rules();//calling Rules()
    cout<<"Enter the number of players(max. 2).\n";
    cin>>n;
    if (n>2)//Size check
      n=2;
  int arr[n];
  int fin[n];//1 if finished, 0 if not.
  var va[n];//array object for counting number of turns, snakes and ladders encountered

       for(int i = 0;i < n;i++)//setting initial positions
        {
         arr[i] = 1;
         fin[i] = 0;
         va[i].nt = 0;
         va[i].ns = 0;
         va[i].nl = 0;
        }//done setting initial positions

  while((n > 0) && (flag < n))//master loop.
        {
        for(int i = 0;i < n;i++)//submaster loop
        {
         if(arr[i] >= 100)
         {
          arr[i] = 100;
          cout<<"Player "<<i+1<<" has finished the game.\n\n";
          continue;
         }

         if(arr[i] < 100 && arr[i] > 0)//outer if
         {
              cout<<"Enter r to roll. Entering anything else will forfeit your turn.\n";
              cout<<"Player "<<(i+1)<<" : ";
              cin>>ch;

        if(ch == 'r')//start of inner if
        {
                roll = cmm.dice(va, i);
                arr[i] = arr[i] + roll;
                va[i].nt++;
                fin[i] = cmm.checkhundred(arr, i);
                cmm.pixelputter(arr, n);
                cout<<"Player "<<i+1<<" rolled a "<<roll<<".\n";
                if(fin[i] == 1)
                    {
                    cout<<"Player "<<i+1<<" has finished the game.\n\n";
                    arr[i] = 100;
                    continue;
                    }
                cout<<"Player "<<i+1<<" is now at position "<<arr[i]<<endl<<endl;
                             if(arr[i] == 49 || arr[i] == 62 || arr[i] == 99)
                             //snake condition
                             {
                              cout<<"SNAKE!!!";
                              va[i].ns++;
                              arr[i] = cmm.demote(arr[i]);
                              cmm.pixelputter(arr,n);
                              cout<<"Player "<<i+1<<" is now at position
"<<arr[i]<<endl<<endl;
                             }
                             if(arr[i] == 2 || arr[i] == 28 || arr[i] == 51)
                             //ladder condition
                             {
                              cout<<"LADDER!!!";
                              va[i].nl++;
                              arr[i] = cmm.promote(arr[i]);
                              cmm.pixelputter(arr,n);
                              cout<<"Player "<<i+1<<" is now at position
"<<arr[i]<<endl<<endl;
                             }
                while(roll == 6)//checking for six
                        {
                        cout<<"Rolling die a second time. Enter r to roll.\n";
                                       cout<<"Player "<<(i+1)<<" : ";
                                       cin>>ch;
                        if (ch == 'r')
                            {
                            roll = cmm.dice(va, i);//bonus roll
                            }
                        arr[i] = arr[i] + roll;
                        fin[i] = cmm.checkhundred(arr, i);
                        cmm.pixelputter(arr,n);
                        cout<<"Player "<<i+1<<" rolled a "<<roll<<".";
                        if(fin[i] == 1)
                            {
                            cout<<"Player "<<i+1<<" has finished the game.\n\n";
                            arr[i] = 100;
                            continue;
                            }
                        cout<<" Player "<<i+1<<" is now at position
"<<arr[i]<<endl<<endl;
                        }//end of checking for six
                        ch = ' ';

        }//end of inner if
        }//end of outer if
        roll = 0;
        }//end of submaster loop
        flag = 0;
        for(int i = 0;i < n;i++)
                        {
                        if(arr[i] >= 100)
                         flag += 1;
                        if(flag == n)
                         break;
                         }


  }//end of master loop

  fstream repoj;
  repoj.open("PlayerRep.dat", ios::out|ios::binary);
  for(int i = 0; i < n;i++)
   repoj.write((char*)&va[i], sizeof(va[i]));
  repoj.close();
  wr = dob.Report(n);
  if(wr != 0)
   cout<<"\nThe Winner is Player "<<wr<<".\n";
  else
   cout<<"\nThe game is tied.\n";



}//end of super if
system("pause");

}//end of main()
```

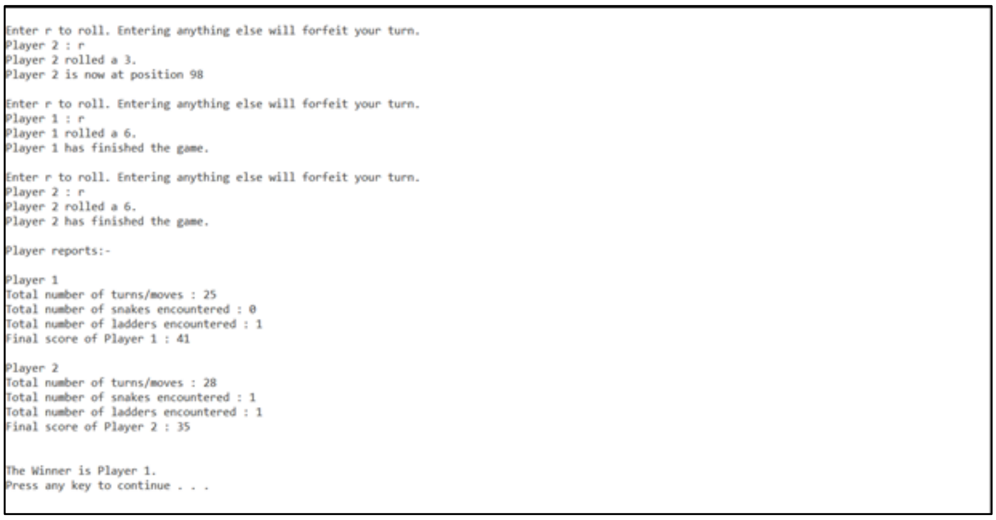
OUTPUT SCREENSHOTS



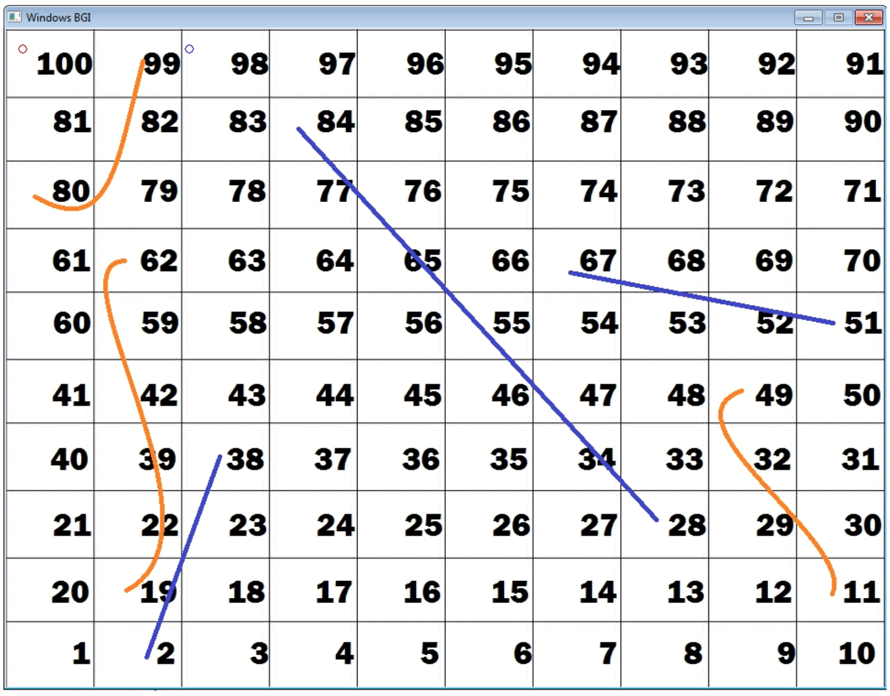




```
Enter y to display the rules and n to not display them.
y
1. The maximum number of players is 2.
2. Enter r to roll.
3. If a die rolls six, the player gets a second roll.
4. Orange represents a snake, and blue represents a ladder.
5. The final score is calculated based on number of turns, number of ladders and snakes encountered.
Enter the number of players(max. 2).
2
Enter r to roll. Entering anything else will forfeit your turn.
Player 1 : r
Player 1 rolled a 6.
Player 1 is now at position 7

Rolling die a second time. Enter r to roll.
Player 1 : r
Player 1 rolled a 3. Player 1 is now at position 10

Enter r to roll. Entering anything else will forfeit your turn.
Player 2 : r
Player 2 rolled a 6.
Player 2 is now at position 7

Rolling die a second time. Enter r to roll.
Player 2 : r
Player 2 rolled a 6. Player 2 is now at position 13

Rolling die a second time. Enter r to roll.
Player 2 : r
Player 2 rolled a 4. Player 2 is now at position 17
```



```
Enter r to roll. Entering anything else will forfeit your turn.
Player 2 : r
Player 2 rolled a 3.
Player 2 is now at position 98

Enter r to roll. Entering anything else will forfeit your turn.
Player 1 : r
Player 1 rolled a 6.
Player 1 has finished the game.

Enter r to roll. Entering anything else will forfeit your turn.
Player 2 : r
Player 2 rolled a 6.
Player 2 has finished the game.

Player reports:-

Player 1
Total number of turns/moves : 25
Total number of snakes encountered : 0
Total number of ladders encountered : 1
Final score of Player 1 : 41

Player 2
Total number of turns/moves : 28
Total number of snakes encountered : 1
Total number of ladders encountered : 1
Final score of Player 2 : 35


The Winner is Player 1.
Press any key to continue . . .
```



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

SCOPE FOR IMPROVEMENT

- The fun and thrill of throwing a die in a game of Snakes and ladders is an integral part of playing the game. Instead of just asking the user to enter ‘r’ to roll, a button complete with a die performing rolling action could have been added.

- The addition of an all-time leader board could have given a more competitive and real world feature to the project and the game.

- Considering the program, a lot more of the code could have been added into functions and classes, thus saving memory and time.

- Instead of allowing the player to directly be able to reach 100 no matter how much in excess his/her die rolls, the program could forfeit the turn until the right number to reach 100 is rolled.

- The snakes and ladders could have been more visually appealing instead of just being plain, coloured lines and curves.

REFERENCES

I referred to the following sources while working on this project:-

- Bibliography

    1. Sumita Arora – Computer Science with C++ Textbook XII Volume I 10th Edition
    2. APC – Understanding Computer Applications Grade X
    3. Sumita Arora – Computer Applications ICSE Class X

- Webliography

    1. stackoverflow.com
    2. sourcecodesworld.com
    3. sanfoundry.com
    4. codechef.com