INDEX

BONAFIDE CERTIFICATE

Certified to the Bonafide Report of work done by Master of Class ***<u>XII</u>*** in Sri Sankara Senior Secondary School, Adyar, Chennai-20

During the year **<u>2016-17</u>**

Dated _____________ Subject Teacher

Submitted for All-India Higher Secondary Practical Examination held in **Computer Science** at **Sri Sankara Senior Secondary School, Adyar, Chennai-20**

External Examiner

Dated __________ Seal

ACKNOWLEDGEMENT

We would like to express our gratitude and deep regards to our teachers **Mrs. S Vidyalakshmi** and **Mrs. Revathy** for their guidance and encouragement given throughout the course of this project entitled **Game Hub.**

We further thank our Principal **Mrs. Mita Venkatesh**, and the staff members of **Sri Sankara Senior Secondary School**, for the apparatus and valuable information provided by them.

We are obliged to our parents and our classmates for helping us finalize this project report.

FLAPPY FISH

About the game:

Flappy Fish is a game where the player (user) must move a fish vertically on the screen, to avoid colliding with moving rectangular bars.

The fish is placed in the left-hand side of the screen, while the bars, which are   units tall and   units wide move from the right hand side of the screen to the left hand side of the screen. The bars have 40 units of space between them, through which the fish can pass unharmed.

The main aim of the game is to last as long as possible without getting out. The user gains a point every second. The user is out once any part of the fish makes contact with any bar. Each set of bars also move vertically, making it a challenge for the user to not get out quickly.

Bonus Points:

There are also a set of bonus points which appear occasionally. They include the small circle, big circle and the square. Colliding with the small circle increments score by 100 points, colliding with the big circle increments the score by 50 points whereas colliding with the square allows the user to remain not out even after colliding with the bars for a specific time.

How the game was coded:

The fish was drawn by using a bunch of line functions. The bars were generated by drawing 4 pairs of 2 bars each using bar function. The initial position of the bars are randomized, after which the x coordinates of the bar are decremented with time so that the bars move from the right to the left of the screen with the passage of time. When the bars move to the extreme left, they again automatically start from the right of the screen. The game uses a function called kbhit() which takes an input without stopping the game, and the game uses this function to keep the bars moving even when the user is not pressing any key. The collision between the fish and the bars to signal the end of the game is done by checking each line of the fish with each line of the bar. In turn, each line of the fish is compared with each line of the bar by using a for loop which compares coincidence of each point on the fish with each point on the bar. Colliding brings the game to an end immediately. The same concept is used for checking collision between the fish and the bonus points also. The bonus points appear in the screen and are governed by a probability function. Using probability, the probability of occurrence of a square is less than probability of occurrence of small circle is less than the probability of occurrence of big circle. The score increments by 1 each second, and on collision with the bonus points, increase as it should. After the game ends, the score and the name of the person are stored in a class highscore which is inserted in a binary file, and also compares with previous scores stored in the binary file. The highest score along with the name of the person is displayed.

Functions used

Collision()- It is a user defined function which checks points of coincidence between each line of the fish and each line of the bar by using a for loop. The initial and final points of the for loop were calculated by calculating the slope of each line.

Collisioncircle1()- It is a user defined function. It checks for points of coincidence between each line of the fish and each point of the small circle (bonus point) by using a for loop.

Collisioncircle2()- It is a user defined function. It checks for points of coincidence between each line of the fish and each point of the big circle (bonus point) by using a for loop.

Drawfish()- It uses 7 line functions to draw the fish.

Collisionsquare()- It is a user defined function. It checks for point of coincidence between each line of the fish and each line of the square (bonus point) by using a for loop.

Kbhit()- It is pre-defined function which does not halt the program but still takes input from the user. This function was used in the game to keep the bars constantly moving without waiting for the user to move the fish. This is the most integral part of the whole game.

CATCH 22

About the game:

Catch 22 is a game where the player (hero) must catch a villain in a limited number of moves.

The hero and the villain are placed in a 10x10 square grid, within which they are free to move about. They both can move only one square at a time, either vertically or horizontally, but not diagonally. The user can move the hero up or down using the W and S keys respectively, and left or right using the A and D keys respectively.

The hero is represented as a red circle, while the villain is represented as a white circle. Initially, the hero and villain are positioned at opposite corners of the grid. The hero then has 22 moves in which to catch the villain. Basically, the user must chase the villain until the hero and the villain end up on the same square. When this happens, the user has won the game.

However, there is a catch;

While the hero cannot go out of the grid under any circumstances, the villain has a special ability: he can move out of one edge of the grid and reappear on the opposite edge. This makes it nearly impossible to corner the villain, as he can disappear from any edge of the grid.

How the game was coded:

The code for catch 22 incorporates graphics along with functions, which essentially serve the purpose of positioning the hero and the villain.

Inside main(), a while loop continuously gets a character input from the user, which is used to move the hero. The input is sent to the functions intposx and intposy, which retutn the hero's position co-ordinates, which are passed on to void heropos(), which draws the hero, if he is within the grid. The villain's movement is determined using a random function within main. His new positon co-ordinates are passes by reference to the function villain(), which modifies his position co-ordinates if needed. The villain is then drawn using heropos(). The positon co-ordinates of the hero and villain are now compared to check if the user has won. All this is done inside a while loop, as the number of moves also need to be counted.

Functions used:

- void grid () – This function draws the grid on the screen. It draws ten vertical and horizontal white lines in the centre of the screen. The grid consists of 10x10 small squares.
- void heropos (int a, int b) – This function accepts two integers which determine the position of the hero. It then draws a circle at the required positon. This same function is called separately in the main to draw both the hero and the villain.
- int posx(char a) – This function returns the X co-ordinate of the hero based on the input (a) from the user.
- int posy(char a) – This function returns the Y co-ordinate of the hero based on the input (a) from the user.
- void villain(int&vilposx, int&vilposy) – This function basically serves the purpose of determining the position co-ordinates of the villain. It checks if the villain is positioned inside the grid or not. If the villain is outside the grid, it repositions him on the opposite edge of the grid by changing the values o vilposx and vilposy accoridingly.

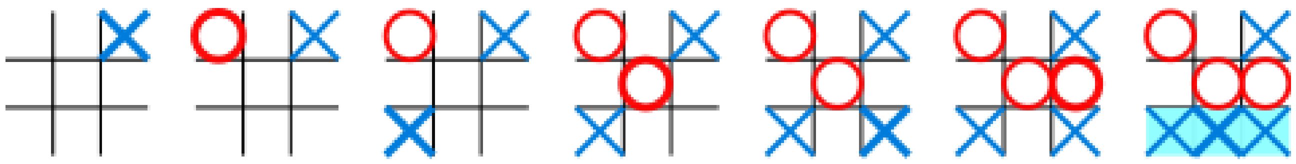
TIC-TAC-TOE

About the game:

Tic-tac-toe is a game for two players, *X* and *O*, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

The rules of the game are very simple. Since the goal is to get three in a row, the players must take turns, first X then O, or vice-versa.

The following is an example game which is begun and won by X:



In the program, the boxes are numbered one to nine, similar to how numbers on a cell phone are placed. The user starts the game and plays X while the computer plays O.

How the game was coded:

The program uses structures, functions and graphics.
A grid is first drawn using the function void dgrid(). The user then plays first. If the user plays in the center square, the computer plays in any one of the corners. If the user plays anywhere else, the computer plays in the center. After the user's second move, the functions winpredict2() is called to see if the user has a chance of winning in the third move. Accordingly, the computer plays in order to prevent that from happening. After this, winpredict1(), winpredict2() and wincheck() are called after each and every move to see if either the computer or the player has a chance of winning or if the game has already been won by either the player of the user. The function boxcheck() is called after each and every move so that filled boxes don't get filled. If the player plays in a filled, the message

“square is filled” will be displayed on screen. All this happens inside a while loop, which runs provided the game hasn’t ended. The result of the game is also displayed using graphics.
This game was originally thought to be unbeatable, but actually has just 4 sequences of moves that can beat it. Thus, the real challenge lies in beating the game!

Structures Used:

- *struct box:* Consists of ‘check’ and ‘type’ variables of data-type integer which store values which help determine if the box is filled and if it is filled by an ‘O’ or ‘X’ respectively.
- *struct grid1:* Consists of structure array of type box and array of size 9 for storing information about the status of each box.

Functions Used:

- *void dgrid():* This function draws the 3x3 grid, of 9 squares, on which the game is played out.
- *void draw():* Invoked when the game results in a draw to display the result as ‘DRAW’ using the existing line() function.
- *void loss():* Invoked when the game results in a loss for the user to display the result as ‘LOSS’ using the existing line() function.
- *void win():* Invoked when the game results in a win for the user to display the result as ‘WIN’ using the existing line() function.
- *void drawo(int a):* Used to place an ‘O’ when the computer makes a move in a specific box which is identified by the integer a, which is the number of the box.
- *void drawx(int a):* Used to place an ‘X’ when the player makes a move in a specific box which is identified by the integer a, which is the number of the box.
- *int boxcheck(grid1 g, int a):* This function is used to check if a box is filled or not. The parameter of type grid1 is used to

store/check the status of a particular box and the one of type integer is used to determine which box.

- *int wincheck(grid1 g):* This function is used to check if the game has ended at any point of time. Invoked only after 3 moves.
- int winpredict1(grid1 g): This function is used to calculate if the computer has a chance to win the game on its next move at any point in time, and then that move is made based on the value returned by this function.
- *int winpredict2(grid1 g):* This function is used to calculate if the player has a chance of winning the game on their next move at any point in time and then that move is prevented by the computer, which plays X in a particular square. The number of the square where the move is to be played is returned by the function.

Integration of the three games into one

On running the game, it asks for a choice from the user on which game the user wants to play. This window is called the main menu. The functions of all the games were copied into the main game whereas the void main() part of all the three games were copied into if conditions. Depending on the choice of the user in the beginning, each if condition is implemented, ie, each game is implemented. On ending of each game, another choice is given to the user on whether he wants to play the same game again, or go to the main screen. Each game is put in a do while loop. If the user wants to play the game again, the do while loop is run again whereas if he wants to go to the main menu, a goto statement takes the program back to the main menu. This way, the user can play whichever game he wants, how many ever times he wants to. There is also a choice in the main menu for quitting the program.

THE CODE

```
#include <fstream.h>
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <string.h>
/* This game is called the flappy fish. Here the fish can be moved up
and down using 'w' and 's', and cannot be moved sideways. The bars will
continuosly move towards you with gaps through which the fish should move.
Colliding with the bars will mean the end of the game. The score increments
with the time the fish is alive. There are bonus coins which increment the
score faster. There are 5 functions used in this program, one for detecting
collision between the fish and the bars, 3 for detecting collision between
fish and the three bonus coins and the last function for drawing the bird.
Class is used for inputting the user's name, username and password whereas
files is used for storing password and highscores. Graphics is used
throughout the game. The game has a tutorial. */
* function for detecting collision between bars and fish. This function
compares each side of the bar with each side of the fish. The parameters
'a' is left of bar, 'b' is top of bar, 'c' is right of bar, 'd' is bottom
of bar, 'y' is y axis of fish, 'z' is x axis of fish. */
class highscore
{
  int score;
  char name[20];
  public:
  int returnscore()
  {
    return score;
  }
  void input(int s, char n[])
  {
    score=s;
    strcpy(name,n);
  }
  void display()
  {
    cout<<name<<" "<<score<<endl;
  }
  highscore()
  {
    score=0;
    strcpy(name," ");
  }
};
int collision(int a, int b, int c, int d, int y, int z)
{
  int flag=0, j, i, k;
  // comparing left of bar with bottom right most of fish //
  for(i=b; i<=d; i++)
  {
     for(j=z, k=y; j>=z-20; j--, k++)
     {
       if(j==a && k==i)
       {
             flag=1;
             goto collision;
       }
     }
  }
```

```
// comparing bottom of bar with bottom right most of fish //
for(i=a; i<=c; i++)
{
  for(j=z, k=y; j>=z-20; j--, k++)
   {
     if(j==i && k==d)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing top of bar with bottom right most of fish //
for(i=a; i<=c; i++)
{
  for(j=z, k=y; j>=z-20; j--, k++)
   {
     if(j==i && k==b)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing left of bar with top right most of fish //
for(i=b; i<=d; i++)
{
  for(j=z, k=y; j>=z-20; j--, k--)
  {
    if(j==a && k==i)
     {
           flag=1;
```

```
          goto collision;
      }
    }
}
// comparing bottom of bar with top right most of fish //
for(i=a; i<=c; i++)
{
  for(j=z, k=y; j>=z-20; j--, k--)
   {
     if(j==i && k==d)
      {
          flag=1;
          goto collision;
      }
    }
}
// comparing top of bar with top right most of fish //
for(i=a; i<=c; i++)
{
  for(j=z, k=y; j>=z-20; j--, k--)
   {
     if(j==i && k==b)
      {
          flag=1;
          goto collision;
      }
    }
}
// comparing left of bar with top middle right of fish //
for(i=b; i<=d; i++)
{
  for(j=z-20, k=y-20; j>=z-50; j-=3, k++)
  {
    if(j==a && k==i)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing bottom of bar with top middle right of fish //
for(i=a; i<=c; i++)
{
  for(j=z-20, k=y-20; j>=z-50; j-=3, k++)
   {
     if(j==i && k==d)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing top of bar with top middle right of fish //
for(i=a; i<=c; i++)
{
  for(j=z-20, k=y-20; j>=z-50; j-=3, k++)
   {
     if(j==i && k==b)
     {
           flag=1;
           goto collision;
     }
   }
}
```

```
// comparing left of bar with bottom middle right of fish //
for(i=b; i<=d; i++)
{
  for(j=z-20, k=y+20; j>=z-50; j-=3, k--)
  {
    if(j==a && k==i)
     {
            flag=1;
            goto collision;
     }
   }
}
// comparing bottom of bar with bottom middle right of fish //
for(i=a; i<=c; i++)
{
  for(j=z-20, k=y+20; j>=z-50; j-=3, k--)
   {
     if(j==i && k==d)
     {
            flag=1;
            goto collision;
     }
   }
}
// comparing top of bar with bottom middle right of fish //
for(i=a; i<=c; i++)
{
  for(j=z-20, k=y+20; j>=z-50; j-=3, k--)
   {
     if(j==i && k==b)
     {
            flag=1;
```

```
          goto collision;
     }
   }
}
// comparing left of bar with top middle left of fish //
for(i=b; i<=d; i++)
{
  for(j=z-50, k=y-10; j>=z-60; j--, k--)
  {
    if(j==a && k==i)
     {
          flag=1;
          goto collision;
     }
   }
}
// comparing bottom of bar with top middle left of fish //
for(i=a; i<=c; i++)
{
  for(j=z-50, k=y-10; j>=z-60; j--, k--)
   {
     if(j==i && k==d)
     {
          flag=1;
          goto collision;
     }
   }
}
// comparing top of bar with top middle left of fish //
for(i=a; i<=c; i++)
{
  for(j=z-50, k=y-10; j>=z-60; j--, k--)
```

```
   {
     if(j==i && k==b)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing left of bar with bottom middle left of fish //
for(i=b; i<=d; i++)
{
  for(j=z-50, k=y+10; j>=z-60; j--, k++)
  {
    if(j==a && k==i)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing bottom of bar with bottom middle left of fish //
for(i=a; i<=c; i++)
{
  for(j=z-50, k=y+10; j>=z-60; j--, k++)
   {
     if(j==i && k==d)
     {
           flag=1;
           goto collision;
     }
   }
}
```

```
// comparing top of bar with bottom middle left of fish //
for(i=a; i<=c; i++)
{
  for(j=z-50, k=y+10; j>=z-60; j--, k++)
   {
     if(j==i && k==b)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing left of bar with left of fish //
for(i=b; i<=d; i++)
{
  for(j=z-60, k=y-20; k<=y+20; k++)
  {
    if(j==a && k==i)
     {
           flag=1;
           goto collision;
     }
   }
}
// comparing bottom of bar with left of fish //
for(i=a; i<=c; i++)
{
  for(j=z-60, k=y-20; k<=y+20; k++)
   {
     if(j==i && k==d)
     {
           flag=1;
```

```
            goto collision;
      }
    }
  }
  // comparing top of bar with left of fish //
  for(i=a; i<=c; i++)
  {
    for(j=z-60, k=y-20; k<=y+20; k++)
    {
      if(j==i && k==b)
      {
            flag=1;
            goto collision;
      }
    }
  }
  collision:
  return flag;
}
/* detecting collision between fish and small circle (bonus) where b is
y axis of fish and y is center of circle of radius 5 */
int collisioncircle(int y, int b)
{
  int flagcollisioncircle=0;
  if((b-20>=y-5 && b-20<=y+5) || (b+20>=y-5 && b+20<=y+5))
  flagcollisioncircle=1;
  return flagcollisioncircle;
}
/* detection of collision between fish and large circle (bonus) where 'b' is
y axis of fish and 'y' is center of circle with radius 10 */
int collisioncircle2(int y, int b)
{
   int flagcollisioncircle=0;
  if((b-20>=y-10 && b-20<=y+10) || (b+20>=y-10 && b+20<=y+10))
  flagcollisioncircle=1;
  return flagcollisioncircle;
}
/* function to draw fish getting coordinates of right most point (a,b) as
parameters */
void drawfish(int a, int b)
{
   setcolor(6);
   line(a,b,a-20,b+20);
   line(a-20,b+20,a-50,b+10);
   line(a-50,b+10,a-60,b+20);
   line(a-60,b+20,a-60,b-20);
   line(a-60,b-20,a-50,b-10);
   line(a-50,b-10,a-20,b-20);
   line(a-20,b-20,a,b);
}
int collisionsquare(int x, int b)
{
  int flag=0;
  if((b-20<=x && b-20>=x-10) || (b+20<=x && b+20>=x-10))
  flag=1;
  return flag;
}
//^ flappy fish
//catch22
int x=1,y=1;
//function to move villain
void villain(int&vilposx,int&vilposy)
{
  int i= 5+random(16);
  if(i%4==0)
    vilposx--;
  else if(i%4==1)
    vilposx++;
  else if(i%4==2)
    vilposy--;
  else
    vilposy++;
  if(vilposx>10)
    vilposx=1;
  if(vilposx<1)
    vilposx=10;
  if(vilposy>10)
    vilposy=1;
  if(vilposy<1)
    vilposy=10;
}
//function to draw grid
void grid()
{
  for(int i=0;i<=10;i++)
  {
    line(50,50+40*i,450,50+40*i);
    line(50+40*i,50,50+40*i,450);
  }
}

 //function to draw hero
void heropos(int a,int b)
{
  if(a>=1 && a<=10 && b>=1 && b<=10)
  {
    for (int i=1;i<15;i++)
      circle(30+40*a,30+40*b,i);
  }
  else
  exit(0);
}

/*functions to check input and accordingly change x and y co-ordinates of
hero. These two functions return value of x and y to main*/
int posx(char a)
{
  if(a==97)
    x--;
  else
    x++;
  return x;
}
int posy(char a)
{
  if(a==115)
    y++;
  else
    y--;
  return y;
}

//tictactoe
void draw()
{
  line(450,300,490,300);
  line(450,300,450,370);
  line(490,300,490,370);
  line(450,370,490,370);
  line(500,300,500,370);
  line(500,300,540,300);
  line(540,300,540,330);
  line(540,330,500,330);
  line(500,330,540,370);
  line(550,370,570,300);
  line(570,300,590,370);
  line(600,300,610,370);
  line(610,370,620,330);
  line(620,330,630,370);
  line(630,370,640,300);
}
void win()
{
  line(450,300,460,370);
  line(460,370,470,330);
  line(470,330,480,370);
  line(480,370,490,300);
  line(500,300,500,370);
  line(510,300,510,370);
  line(510,300,550,370);
  line(550,300,550,370);
}
void loss()
{
  line(450,300,450,370);
  line(450,370,490,370);
  line(500,300,500,370);
  line(540,300,540,370);
  line(500,300,540,300);
  line(500,370,540,370);
  line(550,300,590,300);
  line(550,300,550,335);
  line(550,335,590,335);
  line(590,335,590,370);
  line(590,370,550,370);
  line(600,300,640,300);
  line(600,300,600,335);
  line(600,335,640,335);
  line(640,335,640,370);
  line(640,370,600,370);
}
  struct box
{
  int check,type;
};

struct grid1
{
  box b[9];
}g;

void drawo(int a)
{
  if(a==0)
    circle(65,75,20);
  if(a==1)
    circle(115,75,20);
  if(a==2)
    circle(165,75,20);
  if(a==3)
    circle(65,125,20);
  if(a==4)
```

```
    circle(115,125,20);
  if(a==5)
    circle(165,125,20);
  if(a==6)
    circle(65,175,20);
  if(a==7)
    circle(115,175,20);
  if(a==8)
    circle(165,175,20);
}


void drawx(int a)
{
  if(a==0)
  {
    line(40,50,90,100);
    line(40,100,90,50);
  }
  if(a==1)
  {
    line(90,50,140,100);
    line(90,100,140,50);
  }
  if(a==2)
  {
    line(140,50,190,100);
    line(140,100,190,50);
  }
  if(a==3)
  {
    line(40,100,90,150);
    line(40,150,90,100);
  }
  if(a==4)
  {
    line(90,100,140,150);
    line(140,100,90,150);
  }
  if(a==5)
  {
    line(140,100,190,150);
    line(140,150,190,100);
  }
  if(a==6)
  {
    line(40,150,90,200);
    line(40,200,90,150);
  }
  if(a==7){line(90,200,140,150);line(90,150,140,200);}
  if(a==8){line(140,150,190,200);line(140,200,190,150);}
}


int boxcheck(grid1 g,int a)
{
  if(g.b[a].check==1)
    return 0;
  else
    return 1;
}


int wincheck(grid1 g)
```

```
{
  int flag=0;


  //012


  {
    if((g.b[0].check==g.b[1].check) && (g.b[1].check==g.b[2].check) &&
(g.b[0].check==1))
    {
      if((g.b[0].type==g.b[1].type) && (g.b[1].type == g.b[2].type))
      {
           flag=1;
           return (g.b[0].type + 1);
      }
    }
  }
  //036
  {
    if((g.b[0].check==g.b[3].check) && (g.b[3].check==g.b[6].check) &&
(g.b[0].check==1))
    {
      if((g.b[0].type==g.b[3].type) && (g.b[3].type == g.b[6].type))
      {
           flag=1;
           return (g.b[0].type + 1);
      }
    }
  }
  //048
  {
    if((g.b[0].check==g.b[4].check) && (g.b[4].check==g.b[8].check) &&
(g.b[0].check==1))
    {
      if((g.b[0].type==g.b[4].type) && (g.b[4].type == g.b[8].type))
      {
           flag=1;
           return (g.b[0].type+1);
      }
    }
  }
  //147
  {
    if((g.b[1].check==g.b[4].check) && (g.b[4].check==g.b[7].check) &&
(g.b[1].check==1))
    {
      if((g.b[1].type==g.b[4].type) && (g.b[4].type == g.b[7].type))
      {
           flag=1;
           return (g.b[1].type+1);
      }
    }
  }
  //258
  {
    if((g.b[2].check==g.b[5].check) && (g.b[5].check==g.b[8].check) &&
(g.b[2].check==1))
    {
      if((g.b[2].type==g.b[5].type) && (g.b[5].type == g.b[8].type))
      {
           flag=1;
           return (g.b[2].type+1);
      }
    }
  }
  //246
  {
    if((g.b[2].check==g.b[4].check) && (g.b[4].check==g.b[6].check) &&
(g.b[2].check==1))
```

```
    {
      if((g.b[2].type==g.b[4].type) && (g.b[4].type == g.b[6].type))
      {
           flag=1;
           return (g.b[2].type+1);
      }
    }
  }
  //345
  {
    if((g.b[3].check==g.b[4].check) && (g.b[4].check==g.b[5].check) &&
(g.b[3].check==1))
    {
      if((g.b[3].type==g.b[4].type) && (g.b[4].type == g.b[5].type))
      {
           flag=1;
           return (g.b[3].type+1);
      }
    }
  }
  //678
  {
    if((g.b[6].check==g.b[7].check) && (g.b[7].check==g.b[8].check) &&
(g.b[6].check==1))
    {
      if((g.b[6].type==g.b[7].type) && (g.b[7].type == g.b[8].type))
      {
           flag=1;
           return (g.b[6].type+1);
      }
    }
  }
  if(flag!=1)
  return 0;
}

int winpredict1(grid1 g) //checks for possibilities for computer to win
{
  int flag=0;
  if((g.b[0].check == g.b[1].check) && (g.b[0].check ==1)  && (g.b[2].check==0))
  {
    if((g.b[0].type==g.b[1].type) && (g.b[0].type == 0))
    {
      flag=1;
      return 2;
    }
  }
  if((g.b[0].check == g.b[2].check) && (g.b[0].check ==1)  && (g.b[1].check!=1))
  {
    if((g.b[0].type==g.b[2].type) && (g.b[0].type == 0))
    {
      flag=1;
      return 1;
    }
  }
  if((g.b[1].check == g.b[2].check) && (g.b[1].check ==1)  && (g.b[0].check!=1))
  {
    if((g.b[1].type==g.b[2].type) && (g.b[1].type == 0))
    {
      flag=1;
      return 0;
    }
  }
  if((g.b[0].check == g.b[3].check) && (g.b[0].check ==1)  && (g.b[6].check!=1))
  {
```

```
  if((g.b[0].type==g.b[3].type) && (g.b[0].type == 0))
  {
    flag=1;
    return 6;
  }
}
if((g.b[0].check == g.b[6].check) && (g.b[0].check ==1)  && (g.b[3].check!=1))
{
  if((g.b[0].type==g.b[6].type) && (g.b[0].type == 0))
  {
    flag=1;
    return 3;
  }
}
if((g.b[3].check == g.b[6].check) && (g.b[3].check ==1)  && (g.b[0].check!=1))
{
  if((g.b[3].type==g.b[6].type) && (g.b[3].type == 0))
  {
    flag=1;
    return 0;
  }
}
if((g.b[0].check == g.b[4].check) && (g.b[0].check ==1)  && (g.b[8].check!=1))
{
  if((g.b[0].type==g.b[4].type) && (g.b[0].type == 0))
  {
    flag=1;
    return 8;
  }
}
if((g.b[0].check == g.b[8].check) && (g.b[0].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[0].type==g.b[8].type) && (g.b[0].type == 0))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[8].check) && (g.b[4].check ==1)  && (g.b[0].check!=1))
{
  if((g.b[4].type==g.b[8].type) && (g.b[4].type == 0))
  {
    flag=1;
    return 0;
  }
}
if((g.b[1].check == g.b[4].check) && (g.b[1].check ==1)  && (g.b[7].check!=1))
{
  if((g.b[1].type==g.b[4].type) && (g.b[1].type == 0))
  {
    flag=1;
    return 7;
  }
}
if((g.b[7].check == g.b[1].check) && (g.b[7].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[7].type==g.b[1].type) && (g.b[7].type == 0))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[7].check) && (g.b[4].check ==1)  && (g.b[1].check!=1))
{
  if((g.b[4].type==g.b[7].type) && (g.b[4].type == 0))
  {
    flag=1;
    return 1;
  }
}
if((g.b[2].check == g.b[5].check) && (g.b[2].check ==1)  && (g.b[8].check!=1))
{
  if((g.b[2].type==g.b[5].type) && (g.b[2].type == 0))
  {
    flag=1;
    return 8;
  }
}
if((g.b[2].check == g.b[8].check) && (g.b[2].check ==1)  && (g.b[5].check!=1))
{
  if((g.b[2].type==g.b[8].type) && (g.b[2].type == 0))
  {
    flag=1;
    return 5;
  }
}
if((g.b[5].check == g.b[8].check) && (g.b[5].check ==1)  && (g.b[2].check!=1))
{
  if((g.b[5].type==g.b[8].type) && (g.b[5].type == 0))
  {
    flag=1;
    return 2;
  }
}
if((g.b[2].check == g.b[4].check) && (g.b[2].check ==1)  && (g.b[6].check!=1))
{
  if((g.b[2].type==g.b[4].type) && (g.b[2].type == 0))
  {
    flag=1;
    return 6;
  }
}
if((g.b[2].check == g.b[6].check) && (g.b[2].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[2].type==g.b[6].type) && (g.b[2].type == 0))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[6].check) && (g.b[4].check ==1)  && (g.b[2].check!=1))
{
  if((g.b[4].type==g.b[6].type) && (g.b[4].type == 0))
  {
    flag=1;
    return 2;
  }
}
if((g.b[3].check == g.b[4].check) && (g.b[3].check ==1)  && (g.b[5].check!=1))
{
  if((g.b[3].type==g.b[4].type) && (g.b[3].type == 0))
  {
    flag=1;
    return 5;
  }
}
if((g.b[3].check == g.b[5].check) && (g.b[3].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[3].type==g.b[5].type) && (g.b[3].type == 0))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[5].check) && (g.b[4].check ==1)  && (g.b[3].check!=1))
{
  if((g.b[4].type==g.b[5].type) && (g.b[4].type == 0))
  {
    flag=1;
    return 3;
  }
}
if((g.b[6].check == g.b[7].check) && (g.b[6].check ==1)  && (g.b[8].check!=1))
{
  if((g.b[6].type==g.b[7].type) && (g.b[6].type == 0))
  {
    flag=1;
    return 8;
  }
}
if((g.b[6].check == g.b[8].check) && (g.b[6].check ==1)  && (g.b[7].check!=1))
{
  if((g.b[6].type==g.b[8].type) && (g.b[6].type == 0))
  {
    flag=1;
    return 7;
  }
}
if((g.b[7].check == g.b[8].check) && (g.b[7].check ==1)  && (g.b[6].check!=1))
{
    if((g.b[7].type==g.b[8].type) && (g.b[7].type == 0))
    {
      flag=1;
      return 6;
    }
  }
  if(flag==0)
  return 9;
}

int winpredict2(grid1 g)//checks for possibility of user winning
{
  int flag=0;
  if((g.b[0].check == g.b[1].check) && (g.b[0].check ==1)  && (g.b[2].check==0))
  {
    if((g.b[0].type==g.b[1].type) && (g.b[0].type == 1))
    {
      flag=1;
      return 2;
    }
  }
  if((g.b[0].check == g.b[2].check) && (g.b[0].check ==1)  && (g.b[1].check!=1))
  {
    if((g.b[0].type==g.b[2].type) && (g.b[0].type == 1))
    {
      flag=1;
      return 1;
    }
  }
  if((g.b[1].check == g.b[2].check) && (g.b[1].check ==1)  && (g.b[0].check!=1))
  {
    if((g.b[1].type==g.b[2].type) && (g.b[1].type == 1))
  {
    flag=1;
    return 0;
  }
}
if((g.b[0].check == g.b[3].check) && (g.b[0].check ==1)  && (g.b[6].check!=1))
{
  if((g.b[0].type==g.b[3].type) && (g.b[0].type == 1))
  {
    flag=1;
    return 6;
  }
}
if((g.b[0].check == g.b[6].check) && (g.b[0].check ==1)  && (g.b[3].check!=1))
{
  if((g.b[0].type==g.b[6].type) && (g.b[0].type == 1))
  {
    flag=1;
    return 3;
  }
}
if((g.b[3].check == g.b[6].check) && (g.b[3].check ==1)  && (g.b[0].check!=1))
{
  if((g.b[3].type==g.b[6].type) && (g.b[3].type == 1))
  {
    flag=1;
    return 0;
  }
}
if((g.b[0].check == g.b[4].check) && (g.b[0].check ==1)  && (g.b[8].check!=1))
{
  if((g.b[0].type==g.b[4].type) && (g.b[0].type == 1))
  {
    flag=1;
    return 8;
  }
}
if((g.b[0].check == g.b[8].check) && (g.b[0].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[0].type==g.b[8].type) && (g.b[0].type == 1))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[8].check) && (g.b[4].check ==1)  && (g.b[0].check!=1))
{
  if((g.b[4].type==g.b[8].type) && (g.b[4].type == 1))
  {
    flag=1;
    return 0;
  }
}
if((g.b[1].check == g.b[4].check) && (g.b[1].check ==1)  && (g.b[7].check!=1))
{
  if((g.b[1].type==g.b[4].type) && (g.b[1].type == 1))
  {
    flag=1;
    return 7;
  }
}
if((g.b[7].check == g.b[1].check) && (g.b[7].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[7].type==g.b[1].type) && (g.b[7].type == 1))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[7].check) && (g.b[4].check ==1)  && (g.b[1].check!=1))
{
  if((g.b[4].type==g.b[7].type) && (g.b[4].type == 1))
  {
    flag=1;
    return 1;
  }
}
if((g.b[2].check == g.b[5].check) && (g.b[2].check ==1)  && (g.b[8].check!=1))
{
  if((g.b[2].type==g.b[5].type) && (g.b[2].type == 1))
  {
    flag=1;
    return 8;
  }
}
if((g.b[2].check == g.b[8].check) && (g.b[2].check ==1)  && (g.b[5].check!=1))
{
  if((g.b[2].type==g.b[8].type) && (g.b[2].type == 1))
  {
    flag=1;
    return 5;
  }
}
if((g.b[5].check == g.b[8].check) && (g.b[5].check ==1)  && (g.b[2].check!=1))
{
  if((g.b[5].type==g.b[8].type) && (g.b[5].type == 1))
  {
    flag=1;
    return 2;
  }
}
if((g.b[2].check == g.b[4].check) && (g.b[2].check ==1)  && (g.b[6].check!=1))
{
  if((g.b[2].type==g.b[4].type) && (g.b[2].type == 1))
  {
    flag=1;
    return 6;
  }
}
if((g.b[2].check == g.b[6].check) && (g.b[2].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[2].type==g.b[6].type) && (g.b[2].type == 1))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[6].check) && (g.b[4].check ==1)  && (g.b[2].check!=1))
{
  if((g.b[4].type==g.b[6].type) && (g.b[4].type == 1))
  {
    flag=1;
    return 2;
  }
}
if((g.b[3].check == g.b[4].check) && (g.b[3].check ==1)  && (g.b[5].check!=1))
{
  if((g.b[3].type==g.b[4].type) && (g.b[3].type == 1))
  {
    flag=1;
    return 5;
  }
}
if((g.b[3].check == g.b[5].check) && (g.b[3].check ==1)  && (g.b[4].check!=1))
{
  if((g.b[3].type==g.b[5].type) && (g.b[3].type == 1))
  {
    flag=1;
    return 4;
  }
}
if((g.b[4].check == g.b[5].check) && (g.b[4].check ==1)  && (g.b[3].check!=1))
{
  if((g.b[4].type==g.b[5].type) && (g.b[4].type == 1))
  {
    flag=1;
    return 3;
  }
}
if((g.b[6].check == g.b[7].check) && (g.b[6].check ==1)  && (g.b[8].check!=1))
{
  if((g.b[6].type==g.b[7].type) && (g.b[6].type == 1))
  {
    flag=1;
    return 8;
  }
}
if((g.b[6].check == g.b[8].check) && (g.b[6].check ==1)  && (g.b[7].check!=1))
{
  if((g.b[6].type==g.b[8].type) && (g.b[6].type == 1))
```

```
    {
      flag=1;
      return 7;
    }
  }
  if((g.b[7].check == g.b[8].check) && (g.b[7].check ==1)  && (g.b[6].check!=1))
  {
    if((g.b[7].type==g.b[8].type) && (g.b[7].type == 1))
    {
      flag=1;
      return 6;
    }
  }
  if(flag==0)
  return 10;
}

void dgrid()
{
  for(int i=0;i<4;i++)
  {
    line(40,50+50*i,40+150,50+50*i);
    line(40+50*i,50,40+50*i,50+150);
  }
}



void drawgrid(int a, int b)
{
  setcolor(4);
  line(a,b,a+300,b);
  line(a,b+100,a+300,b+100);
  line(a+100,b-100,a+100,b+200);
  line(a+200,b-100,a+200,b+200);
}
void drawgrid1(int a, int b)
{
  line(a,b,a+150,b);
  line(a,b+50,a+150,b+50);
  line(a+50,b-50,a+50,b+100);
  line(a+100,b-50,a+100,b+100);
}
void drawox(int a, char ox[4])
{
  if(strcmpi(ox,"comp")==0)
  {
  if(a==1)
  circle(200,100,30);
  if(a==2)
  circle(300,100,30);
  if(a==3)
  circle(400,100,30);
  if(a==4)
  circle(200,200,30);
  if(a==5)
  circle(300,200,30);
  if(a==6)
  circle(400,200,30);
  if(a==7)
  circle(200,300,30);
  if(a==8)
  circle(300,300,30);
  if(a==9)
```

```
circle(400,300,30);
}
else
{
if(a==1)
{
  line(170,70,230,130);
  line(170,130,230,70);
}
if(a==2)
{
  line(270,70,330,130);
  line(270,130,330,70);
}
if(a==3)
{
  line(370,70,430,130);
  line(370,130,430,70);
}
if(a==4)
{
  line(170,170,230,230);
  line(170,230,230,170);
}
if(a==5)
{
  line(270,170,330,230);
  line(270,230,330,170);
}
if(a==6)
{
  line(370,170,430,230);
```

```
    line(370,230,430,170);
  }
  if(a==7)
  {
    line(170,270,230,330);
    line(170,330,230,270);
  }
  if(a==8)
  {
    line(270,270,330,330);
    line(270,330,330,270);
  }
  if(a==9)
  {
    line(370,270,430,330);
    line(370,330,430,270);
  }
  }
}


int main(void)
{

   /* request auto detection */
   int gdriver = DETECT, gmode, errorcode;

   /* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

   /* read result of initialization */
   errorcode = graphresult();
```

```
/* an error occurred */
if (errorcode != grOk)
{
   printf("Graphics error: %s\n", grapherrormsg(errorcode));
   printf("Press any key to halt:");
   getch();
   exit(1);
}
setcolor(4);
line(0,100,0,200);
line(0,200,60,200);
setcolor(2);
line(70,200,100,100);
line(100,100,130,200);
setcolor(4);
line(140,100,140,200);
line(140,200,200,200);
setcolor(2);
line(210,200,240,100);
line(240,100,270,200);
setcolor(3);
line(280,100,280,200);
line(340,100,340,200);
line(280,100,340,100);
line(280,150,340,150);
line(280,200,340,200);
setcolor(3);
line(350,100,350,200);
line(410,100,410,200);
line(350,100,410,100);
line(350,200,410,200);
setcolor(3);
```

```
line(420,100,480,100);
line(450,100,450,200);
setcolor(3);
line(490,100,550,100);
line(490,100,490,150);
line(490,150,550,150);
line(550,150,550,200);
line(550,200,490,200);
setcolor(5);
line(150,250,210,250);
line(180,250,180,350);
line(220,250,280,250);
line(220,300,280,300);
line(220,350,280,350);
line(220,250,220,350);
line(290,250,350,250);
line(290,250,290,350);
line(290,350,350,350);
line(360,250,360,350);
line(420,250,420,350);
line(360,300,420,300);
delay(5000);
        returnmainmenu:
cleardevice();
char mainchoice;
char l;
outtextxy(100,100,"1.Flappy fish");
outtextxy(100,200,"2.Catch 22");
outtextxy(100,300,"3.Tic Tac Toe");
outtextxy(100,400,"4.Exit");
do
{
  mainchoice=getch();
}while(mainchoice!='1' && mainchoice!='2' && mainchoice!='3' && mainchoice!='4');
if(mainchoice=='1')
{
cleardevice();
setcolor(2);
line(100,100,160,100);
delay(200);
line(100,150,160,150);
delay(200);
line(100,100,100,200);
delay(200);
line(170,100,170,200);
delay(200);
line(170,200,230,200);
delay(200);
line(240,200,270,100);
delay(200);
line(270,100,300,200);
delay(200);
line(310,100,370,100);
delay(200);
line(310,140,370,140);
delay(200);
line(310,100,310,200);
delay(200);
line(370,100,370,140);
delay(200);
line(380,100,440,100);
delay(200);
line(380,140,440,140);
delay(200);
```

```
line(380,100,380,200);
delay(200);
line(440,100,440,140);
delay(200);
line(450,100,480,140);
delay(200);
line(480,140,510,100);
delay(200);
line(480,140,480,200);
delay(200);
setcolor(8);
line(140,220,180,220);
delay(200);
line(140,255,180,255);
delay(200);
line(140,220,140,290);
delay(200);
setcolor(4);
line(190,220,190,290);
delay(200);
setcolor(5);
line(200,220,240,220);
delay(200);
line(200,220,200,255);
delay(200);
line(200,255,240,255);
delay(200);
line(240,255,240,290);
delay(200);
line(240,290,200,290);
delay(200);
setcolor(6);
```

```
   line(250,220,250,290);
   delay(200);
   line(290,220,290,290);
   delay(200);
   line(250,255,290,255);
   delay(200);
   drawfish(400,300);
   delay(300);
   clrscr();
  char name[20];
  int fscore;
  cout<<endl<<"Enter your name below"<<endl;
  gets(name);       // inputting name //
  highscore hscore,h1score;
  fstream f("highscore.dat",ios::binary|ios::out);
  f.write((char*)&hscore,sizeof(hscore));
  f.close();
  do
  {
     char c,d,e,f,getch1,getch2; // abbrev. for getch(); //
     float s,r, randomno2, randomno3, randomno4, randomno5, num2, num3, num4,
num5;
     int gdriver = DETECT, gmode, errorcode;
     int jumpsquare, skipcollision, c3, flagskip=0, randomnosquare,score=0,
randomnochoice, randomnocirclechoice, jumpcircle1=0, c2, s1, s2, s3, c1, p1, p2, p3,
p4, p5, p6, p7, p8, midy1, midy2, midx1, midx2, midx3, midx4, a=100, b=200, flag2=0,
flag3=0, flag4=0, flag5=0, flag1=0, randomnocircle, jumpcircle=0;


     /* initialize graphics and local
      variables */
     initgraph(&gdriver, &gmode, "c:\\tc\\bgi");


     /* read result of initialization */
     errorcode = graphresult();
```

```
if (errorcode != grOk)  /* an error occurred */
{
      printf("Graphics error: %s\n", grapherrormsg(errorcode));
      printf("Press any key to halt:");
      getch();
      exit(1);
      /* terminate with an error code */
}

setfillstyle(1, getmaxcolor());

do
{
  clrscr();
  cout<<'\t'<<"Welcome to FLAPPY FISH!!"<<endl<<endl;
  cout<<"Hi ";
  puts(name);
  cout<<endl;
  cout<<"The fish will be as below"<<endl<<endl;
  a=100; b=200;
  drawfish(a,b);     // drawing fish with parameters 100 and 200 //
  cout<<"Press 't' for tutorial or press 1 to play!"<<endl;
  do
  {
       /* getting input of only 't' for tutorial or '1' for playing
       from the user. Input is stored in 'c' */
       c=getch();
       if(c=='t' || c=='1')
       break;
  } while(2>=1);
  if(c=='t')     // tutorial press 't' //
  {
```

```
// first step of tutorial: HOW TO MOVE THE FISH //
do
{
  clrscr();
  cout<<"STEP 1: PRESS 'S' TO MOVE DOWN PRESS 'W' TO MOVE UP"<<endl;
  cout<<"Press 'm' to go to next step"<<endl;
  drawfish(a,b);
  /* getting input of either 'w' to move up or 's' to move down
  or 'm' to go to next step from the user. The input is stored in
  'd' */
  d=getch();
  if(d=='w') // make fish move up //
  {
    b-=2;
    drawfish(a,b);
  }
  else if(d=='s')
  {
    b+=2;
    drawfish(a,b);
  }
} while(d!='m');
// Second step: HOW THE BARS WILL MOVE //
do
{
  clrscr();
  // randomizing the starting coordinates of the bars on top //
  randomize();
  randomno2=random(180.0)+30.0;
  randomno3=random(180.0)+30.0;
  randomno4=random(180.0)+30.0;
  randomno5=random(180.0)+30.0;
```

```
/* getting starting coordinates of bottom bars. Sum of coordinates
of top bar and bttom bar must be 240 */
num2=240.0-randomno2;
num3=240.0-randomno3;
num4=240.0-randomno4;
num5=240.0-randomno5;
int flag10=0;
for(s=50.0, s1=200.0, s2=350.0, s3=500.0; 1<=2; s+=1, s1+=1, s2+=1, s3+=1)
{
  num2=240.0-randomno2;
  num3=240.0-randomno3;
  num4=240.0-randomno4;
  num5=240.0-randomno5;
  // defining y coordinate of bar //
  midy1=getmaxy()-130.0;
  midy2=getmaxy()-430.0;
  /* x coordinate of first bar. If bar reaches the left (s=699)
  then s is reset to 50 (right most) */
  midx1=getmaxx()-s;
  if(s==699)
  s=50.0;
  midx2=getmaxx()-s1;
  if(s1==699)
  s1=50.0;
  midx3=getmaxx()-s2;
  if(s2==699)
  s2=50.0;
  midx4=getmaxx()-s3;
  if(s3==699)
  s3=50.0;
  score+=0.2;
  if(flag2==0)
```

```
{
  if(randomno2<230)
  randomno2+=0.5;
  else
  flag2=1;
}
else
{
  if(randomno2>=30)
  randomno2-=0.5;
  else
  flag2=0;
}
if(flag3==0)
{
  if(randomno3<230)
  randomno3+=0.5;
  else
  flag3=1;
}
else
{
  if(randomno3>=30)
  randomno3-=0.5;
  else
  flag3=0;
}
if(flag4==0)
{
  if(randomno4<230)
  randomno4+=0.5;
  else
  flag4=1;
}
else
{
  if(randomno4>=30)
  randomno4-=0.5;
  else
  flag4=0;
}
if(flag5==0)
{
  if(randomno5<230)
  randomno5+=0.5;
  else
  flag5=1;
}
else
{
  if(randomno5>=30)
  randomno5-=0.5;
  else
  flag5=0;
}

clrscr();
if(kbhit()!=0)
{
      f=getch();
      if(f=='m')
      {
        break;
      }
```

```
            }
            cout<<"STEP 2: THE BARS WILL BE AS SHOWN! NEVER HIT THEM"<<endl;
            cout<<"Press 'm' to go to the next step"<<endl;
            bar(midx1+40, midy1-randomno2, midx1+50, midy1+50);
            bar(midx1+40, midy2-10, midx1+50, midy2+num2);
            bar(midx2+40, midy1-randomno3, midx2+50, midy1+50);
            bar(midx2+40, midy2-10, midx2+50, midy2+num3);
            bar(midx3+40, midy1-randomno4, midx3+50, midy1+50);
            bar(midx3+40, midy2-10, midx3+50, midy2+num4);
            bar(midx4+40, midy1-randomno5, midx4+50, midy1+50);
            bar(midx4+40, midy2-10, midx4+50, midy2+num5);

            delay(25);
          }
          break;
        } while(1<=2);
        do
        {
          clrscr();
          cout<<"STEP 3: SCORE DETAILS"<<endl<<endl<<endl;
          cout<<"Score increases automatically by 1 every second"<<endl<<endl;
          cout<<"Score increases by 50 if you eat this large circle"<<endl<<endl<<endl;
          setcolor(4);
          circle(80,110,10);
          cout<<"Score increases by 100 if you eat this small
circle"<<endl<<endl<<endl;
          setcolor(4);
          circle(80,160,5);
          cout<<"You can hit the bars for a specific time if you eat this
square"<<endl<<endl<<endl;
          setfillstyle(1,4);
          bar(75,200,85,210);
          cout<<"These bonus points occur occasionally"<<endl<<endl;
          cout<<"Press 'm' to return to main menu"<<endl;
          d=getch();
        } while(d!='m');
   }
} while(c!='1');
setfillstyle(1, getmaxcolor());
char q;
if(c=='1')
{
  clrscr();
  do
  {
       cout<<"Press 'w' to continue!! Keep your fingers on w and s!"<<endl;
       q=getch();
  } while(q!='w');
  a=100; b=200;
  randomize();
  randomno2=random(180.0)+30.0;
  randomno3=random(180.0)+30.0;
  randomno4=random(180.0)+30.0;
  randomno5=random(180.0)+30.0;
  num2=240.0-randomno2;
  num3=240.0-randomno3;
  num4=240.0-randomno4;
  num5=240.0-randomno5;

  for(s=50.0, s1=200.0, s2=350.0, s3=500.0; 1<=2; s+=1, s1+=1, s2+=1, s3+=1)
  {

       num2=240.0-randomno2;
       num3=240.0-randomno3;
       num4=240.0-randomno4;
```

```
num5=240.0-randomno5;

midy1=getmaxy()-130.0;
midy2=getmaxy()-430.0;
midx1=getmaxx()-s;
if(s==699)
s=50.0;
midx2=getmaxx()-s1;
if(s1==699)
s1=50.0;
midx3=getmaxx()-s2;
if(s2==699)
s2=50.0;
midx4=getmaxx()-s3;
if(s3==699)
s3=50.0;

if(flag2==0)
{
  if(randomno2<230)
  randomno2+=0.5;
  else
  flag2=1;
}
else
{
  if(randomno2>=30)
  randomno2-=0.5;
  else
  flag2=0;
}
if(flag3==0)
```

```
{
  if(randomno3<230)
  randomno3+=0.5;
  else
  flag3=1;
}
else
{
  if(randomno3>=30)
  randomno3-=0.5;
  else
  flag3=0;
}
if(flag4==0)
{
  if(randomno4<230)
  randomno4+=0.5;
  else
  flag4=1;
}
else
{
  if(randomno4>=30)
  randomno4-=0.5;
  else
  flag4=0;
}
if(flag5==0)
{
  if(randomno5<230)
  randomno5+=0.5;
  else
  flag5=1;
}
else
{
  if(randomno5>=30)
  randomno5-=0.5;
  else
  flag5=0;
}
if((int)s%100==0)
score++;
clrscr();
cout<<"Score : "<<score;
if(kbhit()!=0)
{
          f=getch();
          if(f=='w') // make fish move up //
          {
             b-=2;
             drawfish(a,b);
          }
          else if(f=='s') // make fish move down //
          {
             b+=2;
             drawfish(a,b);
          }
}
else
drawfish(a,b);
setfillstyle(1,2);
bar(midx1+40, midy1-randomno2, midx1+50, midy1+50);
bar(midx1+40, midy2-10, midx1+50, midy2+num2);
bar(midx2+40, midy1-randomno3, midx2+50, midy1+50);
bar(midx2+40, midy2-10, midx2+50, midy2+num3);
bar(midx3+40, midy1-randomno4, midx3+50, midy1+50);
bar(midx3+40, midy2-10, midx3+50, midy2+num4);
bar(midx4+40, midy1-randomno5, midx4+50, midy1+50);
bar(midx4+40, midy2-10, midx4+50, midy2+num5);
if((int)s%400==0)
{
 randomize();
 randomnocirclechoice=random(6);
}
if(randomnocirclechoice==0 || randomnocirclechoice==1)
{
 if((int)s%400==0)
 {
  jumpcircle=0;
  randomize();
  randomnochoice=random(2);
  if(randomnochoice==0)
  {
   randomize();
   randomnocircle=random(b-140)+110;
  }
  else
  {
   randomize();
   randomnocircle=random(345-b)+b+30;
  }
 }
 if((int)s%400>=0 && (int)s%400<=200 && jumpcircle==0)
 {
  setcolor(4);
```

```
            circle(80, randomnocircle,5);
          }
          if(jumpcircle==0)
          {
            c1=collisioncircle(randomnocircle,b);
          }
        }
        else if(randomnocirclechoice==2 || randomnocirclechoice==3 ||
randomnocirclechoice==4)
        {
          if((int)s%400==0)
          {
            jumpcircle1=0;
            randomize();
            randomnochoice=random(2);
            if(randomnochoice==0)
            {
              randomize();
              randomnocircle=random(b-150)+110;
            }
            else
            {
              randomize();
              randomnocircle=random(345-b)+b+40;
            }
          }
          if((int)s%400>=0 && (int)s%400<=200 && jumpcircle1==0)
          {
            setcolor(4);
            circle(80, randomnocircle,10);
          }
          if(jumpcircle1==0)
          {
```

```
    c2=collisioncircle2(randomnocircle,b);
  }
}
else
{
  if((int)s%400==0)
  {
    jumpsquare=0;
    randomize();
    randomnochoice=random(2);
    if(randomnochoice==0)
    {
      randomize();
      randomnosquare=random(b-150)+110;
    }
    else
    {
      randomize();
      randomnosquare=random(345-b)+b+40;
    }
  }
  if((int)s%400>=0 && (int)s%400<=200 && jumpsquare==0)
  {
    setfillstyle(1,4);
    bar(75,randomnosquare-10,85,randomnosquare);
  }
  if(jumpsquare==0)
  {
    c3=collisionsquare(randomnosquare,b);
  }
}
```

```
p1=collision(midx1+40, midy1-randomno2, midx1+50, midy1+50, b, a);
p2=collision(midx1+40, midy2-50, midx1+50, midy2+num2, b, a);
p3=collision(midx2+40, midy1-randomno3, midx2+50, midy1+50, b, a);
p4=collision(midx2+40, midy2-50, midx2+50, midy2+num3, b, a);
p5=collision(midx3+40, midy1-randomno4, midx3+50, midy1+50, b, a);
p6=collision(midx3+40, midy2-50, midx3+50, midy2+num4, b, a);
p7=collision(midx4+40, midy1-randomno5, midx4+50, midy1+50, b, a);
p8=collision(midx4+40, midy2-50, midx4+50, midy2+num5, b, a);
if(p1==1 || p2==1 || p3==1 || p4==1 || p5==1 || p6==1 || p7==1 || p8==1)
{
  break;
}
if(c1==1)
{
  jumpcircle=1;
  score+=100;
}
c1=0;
if(c2==1)
{
  jumpcircle1=1;
  score+=50;
}
c2=0;
if(c3==1)
{
  jumpsquare=1;
  skipcollision=1;
  flagskip=1;
}
c3=0;
float x=25.0;
```

```
        x-=0.1;
        delay(x);
   }
  }
  clrscr();
  int intscore;
  intscore=(int)score;
  cout<<"You lost ";
  fstream f1("highscore.dat",ios::binary|ios::in);
  fstream f2("temp.dat",ios::binary|ios::out);
  h1score.input(intscore,name);
  f1.read((char*)&hscore,sizeof(hscore));
  int highscore1=hscore.returnscore();
  int highscore2=h1score.returnscore();
  if(highscore2>highscore1)
  {
   f2.write((char*)&h1score,sizeof(h1score));
  }
  else
  {
   f2.write((char*)&hscore,sizeof(hscore));
  }
  f1.close();
  f2.close();
  remove("highscore.dat");
  rename("temp.dat","highscore.dat");
  puts(name);
  cout<<endl<<"score: "<<intscore<<endl;
  cout<<"HIGHSCORE"<<endl;
  highscore h2score;
  fstream f3("highscore.dat",ios::binary|ios::in);
  f3.read((char*)&h2score,sizeof(h2score));
```

```
    h2score.display();
    f3.close();
    cout<<"enter 'a' to play again and 'e' to exit"<<endl;
    do
    {
      l=getch();
    } while(l!='a' && l!='e');
  } while(l=='a');
  goto returnmainmenu;
 }
 else if(mainchoice=='2')
 {
   char catch22choice;
   int flagcatch22=0;
   int gdriver = DETECT, gmode, errorcode;
   int vilposx=10,vilposy=10,i=1;char a;

   /* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "c://tc//bgi");

   errorcode = graphresult();
   /* an error occurred */
   if (errorcode != grOk)
   {
     printf("Graphics error: %s\n", grapherrormsg(errorcode));
     printf("Press any key to halt:");
     exit(1);
   }
   setcolor(1);
   line(100,100,160,100);
   line(100,100,100,200);
   line(100,200,160,200);
```

```
delay(200);
setcolor(2);
line(170,200,200,100);
line(200,100,230,200);
delay(200);
setcolor(3);
line(240,100,300,100);
line(270,100,270,200);
delay(200);
setcolor(4);
line(310,100,370,100);
line(310,100,310,200);
line(310,200,370,200);
delay(200);
setcolor(5);
line(380,100,380,200);
line(440,100,440,200);
line(380,150,440,150);
delay(300);
setcolor(6);
line(300,240,400,240);
delay(150);
line(400,240,400,315);
delay(150);
line(400,315,300,315);
delay(150);
line(300,315,300,390);
delay(150);
line(300,390,400,390);
delay(150);
line(420,240,520,240);
delay(150);
```

```
   line(520,240,520,315);
   delay(150);
   line(520,315,420,315);
   delay(150);
   line(420,315,420,390);
   delay(150);
   line(420,390,520,390);
   delay(150);
   do
   {
     flagcatch22=0;
     char flag1;
     cout<<endl<<"Ente r your name below!"<<endl;
     char name[20];
     gets(name);
     clrscr();
     cout<<"WELCOME TO CATCH 22!!"<<endl;
     cout<<"THE RED BALL IS THE HERO"<<endl<<endl<<endl;
     setcolor(RED);
     for(i=1;i<=10;i++)
          circle(350,20,i);
     cout<<"THE WHITE BALL IS THE VILLAIN"<<endl<<endl<<endl;
     setcolor(WHITE);
     for(i=1;i<=10;i++)
          circle(350,60,i);
     cout<<"Use A,S,W,D to move!"<<endl<<endl;
     cout<<"The main objective is to catch the villain, and u have only 22
moves"<<endl<<endl;
     char flag;
     randomize();
     do
     {
          cout<<"press 'm' to continue"<<endl;
```

```
    flag=getch();
} while(flag!='m');

cleardevice();
//block of code for initializing color and displaying grid and hero at (1,1)
{
    setcolor(getmaxcolor());
    grid();
    for(int j=1;j<=15;j++)
    {
      setcolor(RED);
      circle(70,70,j);
      setcolor(getmaxcolor());
      circle(430,430,j);
    }
}
/*block of code for getting input(A,S,D,W). Code has to be written
to compare hero's position with villain's*/
while((x!=vilposx || y!=vilposy) && i<=50)
{
    do
    {
      a=getch();
    }
    while(a!=97 && a!=100 &&a!=115 && a!=119);
    cleardevice();
    i++;
    {
      if(a==97 || a==100)
        x=posx(a);
      else
        y=posy(a);
```

```
          }
          villain(vilposx,vilposy);
          setcolor(RED);
          heropos(x,y);
          setcolor(getmaxcolor());
          heropos(vilposx,vilposy);
          grid();
          if(vilposx==x && vilposy==y)
          {
            circle(30+40*x,30+40*y,20);
            flagcatch22=1;
            break;
          }
     }
     delay(3000);
     cleardevice();
     if(flagcatch22==1)
          outtextxy(50,50," YOU win ");
     else
          outtextxy(50,50," YOU lost ");
          x=y=1;
          vilposx=vilposy=10;
          i=0;
     do
     {
          outtextxy(40+40*x,70+40*y,"'a' to play again and 'e' to exit");
          catch22choice=getch();
     } while(catch22choice!='a' && catch22choice!='e');
  } while(catch22choice=='a');
  goto returnmainmenu;
 }
```

```
else if(mainchoice=='3')
{
  /* request auto detection */
int gdriver = DETECT, gmode, errorcode;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

/* read result of initialization */
errorcode = graphresult();
/* an error occurred */
if (errorcode != grOk)
{
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  getch();
  exit(1);
}
clrscr();

 setcolor(4);
 circle(50,100,30);
 setcolor(5);
 line(20,100,80,100);
 line(50,100,50,130);
 setcolor(1);
 circle(50,85,6);
 delay(200);
 line(90,75,90,130);
 delay(200);
 line(100,75,140,75);
 line(100,75,100,130);
line(100,130,140,130);
delay(500);
setcolor(4);
circle(250,200,30);
setcolor(5);
line(220,200,280,200);
line(250,200,250,230);
setcolor(1);
circle(250,185,6);
delay(200);
line(290,230,310,175);
line(310,175,330,230);
delay(200);
line(340,175,340,230);
line(340,175,380,175);
line(340,230,380,230);
delay(500);
setcolor(4);
circle(450,300,30);
setcolor(5);
line(420,300,480,300);
line(450,300,450,330);
setcolor(1);
circle(450,285,6);
delay(200);
line(490,275,490,330);
line(490,275,530,275);
line(490,330,530,330);
line(530,275,530,330);
delay(200);
line(540,275,540,330);
line(540,275,580,275);
```

```
 line(540,302,580,302);
 line(540,330,580,330);
 delay(200);
 setcolor(8);
 drawgrid1(400,100);
 delay(3000);
 char tictactoechoice;
do
{
  clrscr();
  int a=0,z=0,turn=0;
  for(int i=0;i<9;i++)
  g.b[i].check=0;
  dgrid();

  while( (wincheck(g)==0) && turn<5 )
  {
    a=getch();
    a-=49;
    if(boxcheck(g,a)==1)
    {
         turn++;
         drawx(a);
         g.b[a].type=1;
         g.b[a].check=1;
         {
           if(wincheck(g)!=0)
           break;
         }
         if(a==4 && turn==1)
         {
           int flag=0;
```

```
randomize();
z=random(16);
{
  if(z%4==0)
  {
    if((boxcheck(g,0)==1))
    {
         flag=1;
         z=0;
    }
  }
  else if(z%4==1)
  {
    if((boxcheck(g,2)==1))
    {
         flag=1;
         z=2;
    }
  }
  else if(z%4==2)
  {
    if(boxcheck(g,6)==1)
    {
         flag=1;
         z=6;
    }
  }
  else
  {
    if(boxcheck(g,8)==1)
    {
         flag=1;
```

```
          z=8;
        }
      }
    }
    if(flag==1)
    {
      g.b[z].type=0;
      g.b[z].check=1;
      delay(400);
      drawo(z);
    }
  }
  else if(turn ==1 && a!=4)
  {
    z=4;
    delay(400);
    drawo(z);
    g.b[z].type=0;
    g.b[z].check=1;
  }
  else
  {
    z=winpredict1(g);
    if(z!=9)
    {
      delay(400);
      g.b[z].type=0;
      g.b[z].check=1;
      drawo(z);
    }
    else
    {
            z=winpredict2(g);
            if(z!=10)
            {
              delay(400);
              drawo(z);
              g.b[z].type=0;
              g.b[z].check=1;
            }
            else
            {
              //checked
if((boxcheck(g,0)==1)||(boxcheck(g,2)==1)||(boxcheck(g,6)==1)||(boxcheck(g,8)==1))
              {
                   int flag1=0;
                   while(flag1!=1)
                   {
                     randomize;
                     z=random(20);
                     if(z%4==0)
                     {
                       if(boxcheck(g,0)==1)
                       {
                         flag1=1;
                         delay(400);
                         drawo(0);
                         g.b[0].type=0;
                         g.b[0].check=1;
                         break;
                       }
                     }
                     if(z%4==1)
                     {
```

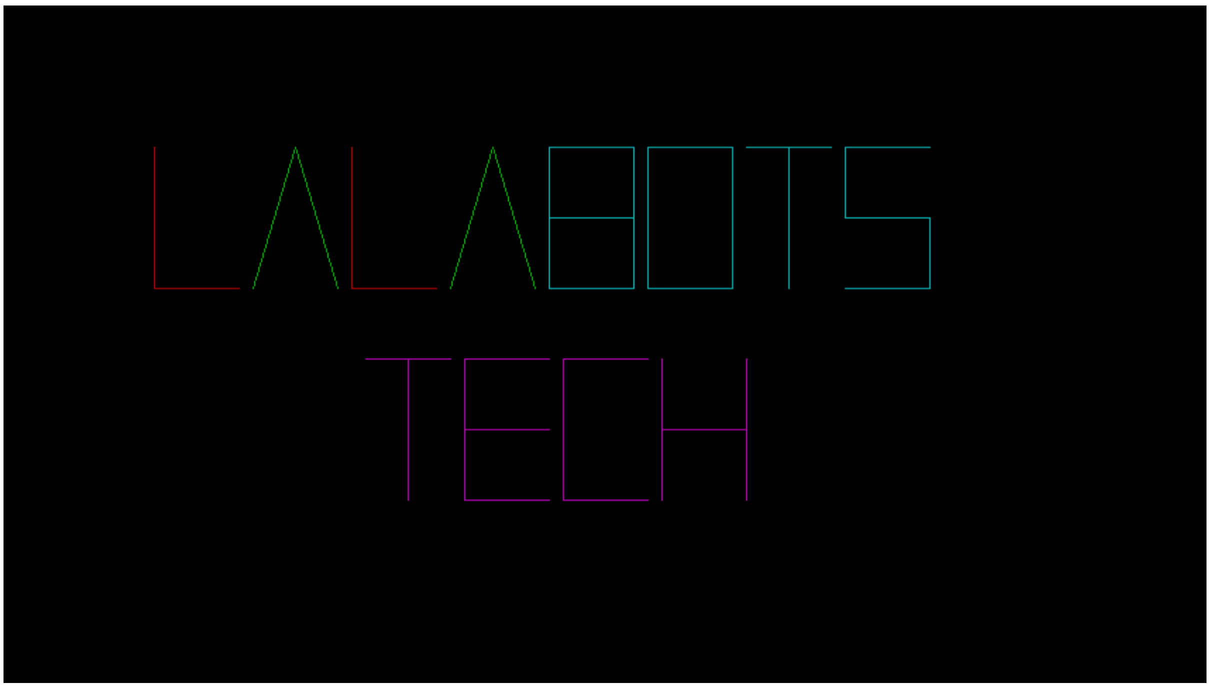
```
  if(boxcheck(g,2)==1)
  {
    flag1=1;
    delay(400);
    drawo(2);
    g.b[2].type=0;
    g.b[2].check=1;
    break;
  }
}
if(z%4==2)
{
  if(boxcheck(g,6)==1)
  {
    flag1=1;
    delay(400);
    drawo(6);
    g.b[6].type=0;
    g.b[6].check=1;
    break;
  }
}
else
{
  if(boxcheck(g,8)==1)
  {
    flag1=1;
    delay(400);
    drawo(8);
    g.b[8].type=0;
    g.b[8].check=1;
    break;
```

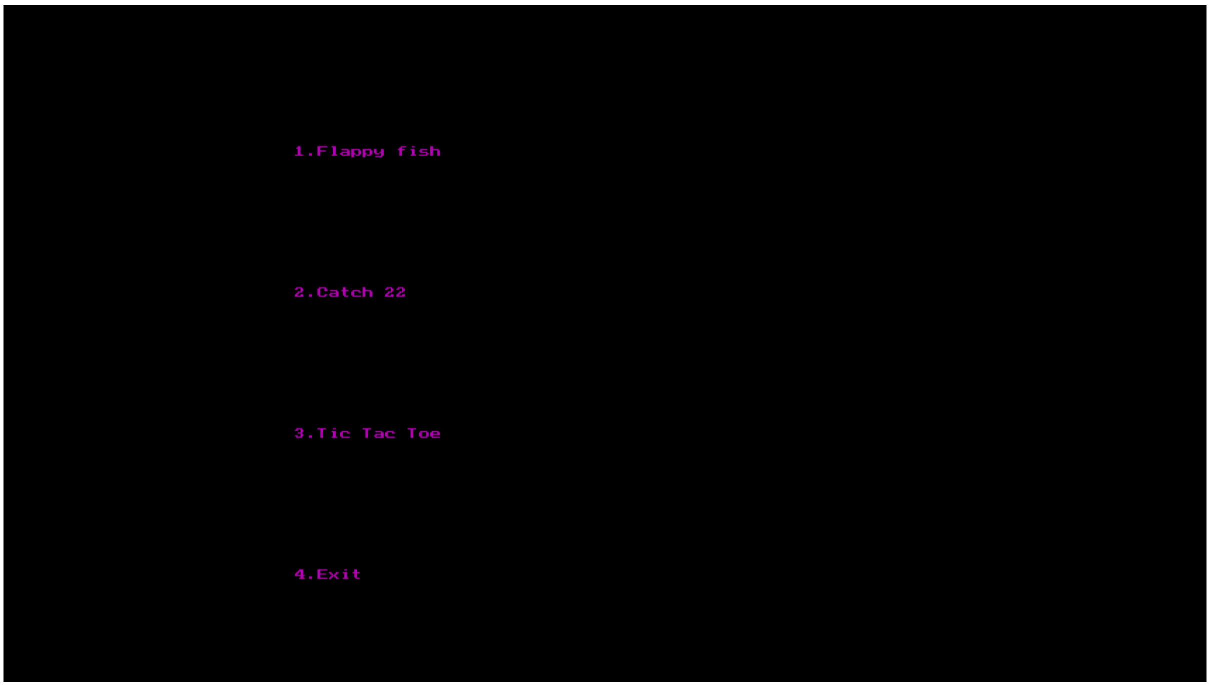
```
                    }
                  }
                }
            }
            else
if((boxcheck(g,1)==1)||(boxcheck(g,3)==1)||(boxcheck(g,5)==1)||(boxcheck(g,7)==1))
            {
                int flag1=0;
                while(flag1!=1)
                {
                  randomize;
                  z=random(20);
                  if(z%4==0)
                  {
                    if(boxcheck(g,1)==1)
                    {
                      flag1=1;
                      delay(400);
                      drawo(1);
                      g.b[1].type=0;
                      g.b[1].check=1;
                      break;
                    }
                  }
                  if(z%4==1)
                  {
                    if(boxcheck(g,3)==1)
                    {
                      flag1=1;
                      delay(400);
                      drawo(3);
                      g.b[3].type=0;
                      g.b[3].check=1;
```

```
          break;
         }
        }
        if(z%4==2)
        {
         if(boxcheck(g,5)==1)
         {
          flag1=1;
          delay(400);
          drawo(5);
          g.b[5].type=0;
          g.b[5].check=1;
          break;
         }
        }
        else
        {
         if(boxcheck(g,7)==1)
         {
          flag1=1;
          delay(400);
          drawo(7);
          g.b[7].type=0;
          g.b[7].check=1;
          break;
         }
        }
       }
    }
    else
    break;
  }//checked
```

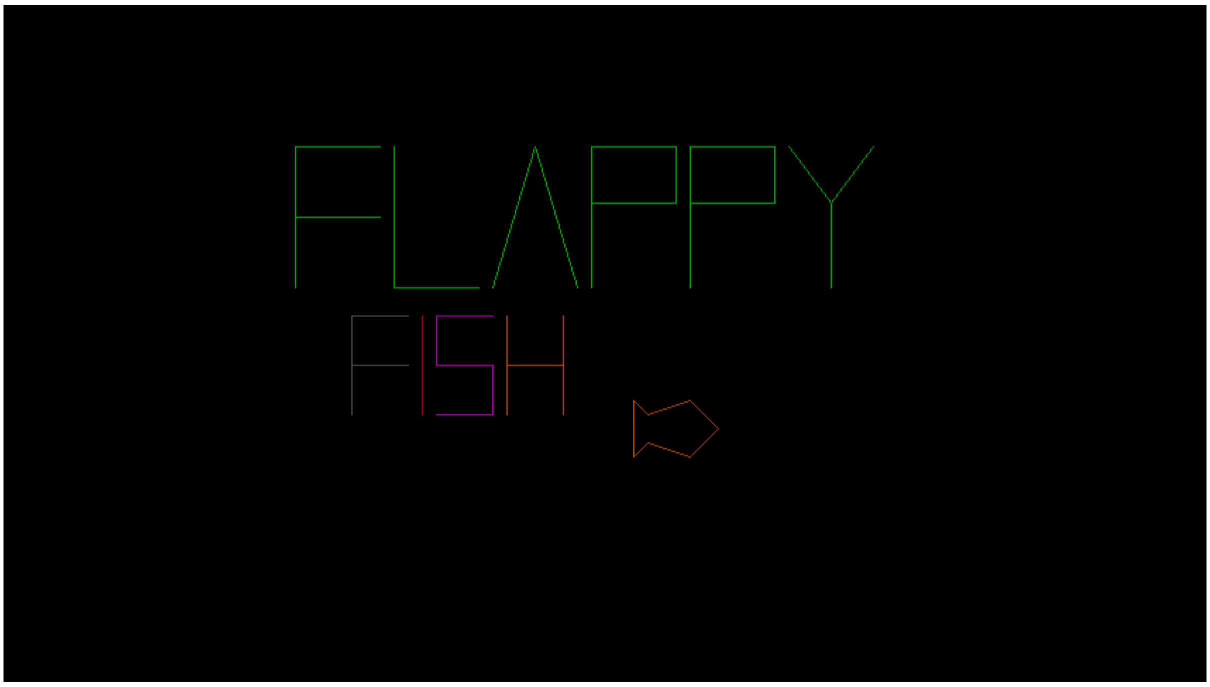
```
           }
         }
     }
     else
         outtextxy(250,350,"square is filled");
   }//correct!!!!
   delay(1000);
   if(wincheck(g)==0)
   {
     cleardevice();
     draw();
   }
   else if(wincheck(g)==1)
   {
     cleardevice();
     loss();
   }
   else
   {
     cleardevice();
     win();
   }
   do
     {
       outtextxy(100,200,"'a' to play again and 'e' to exit");
       tictactoechoice=getch();
     } while(tictactoechoice!='a' && tictactoechoice!='e');
   }while(tictactoechoice=='a');
   goto returnmainmenu;
   }
      /* clean up */
      closegraph();
      return 0;
}
```

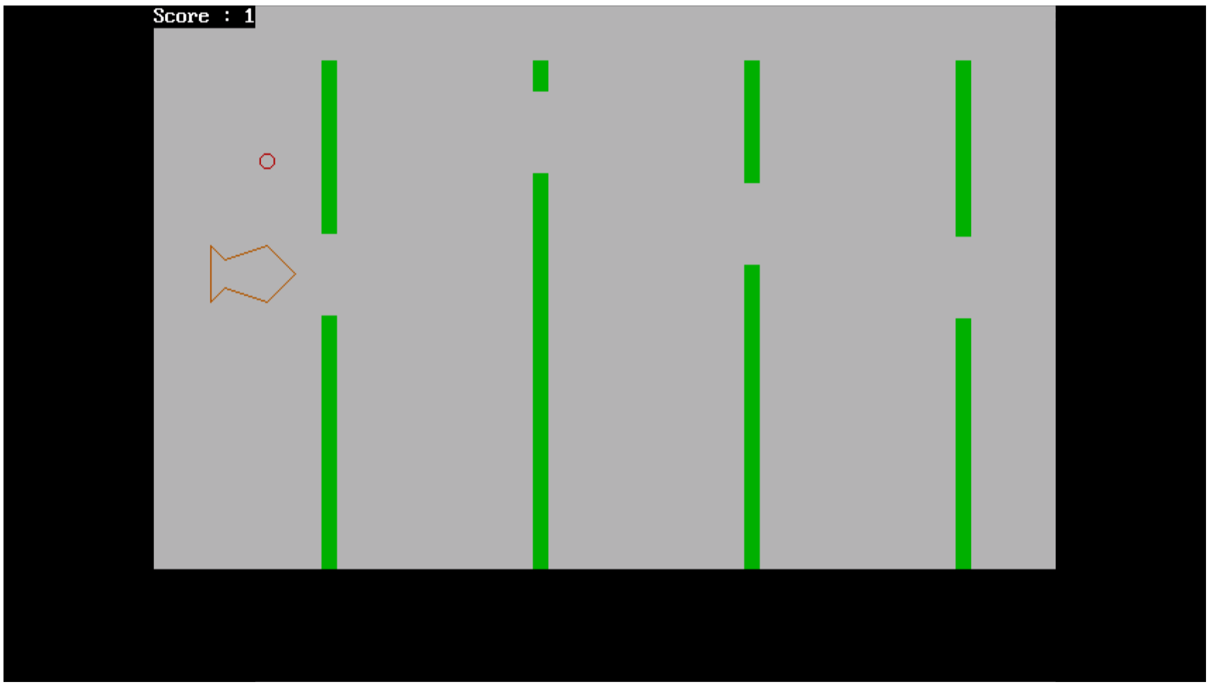
SCREENSHOTS OF THE PROJECT

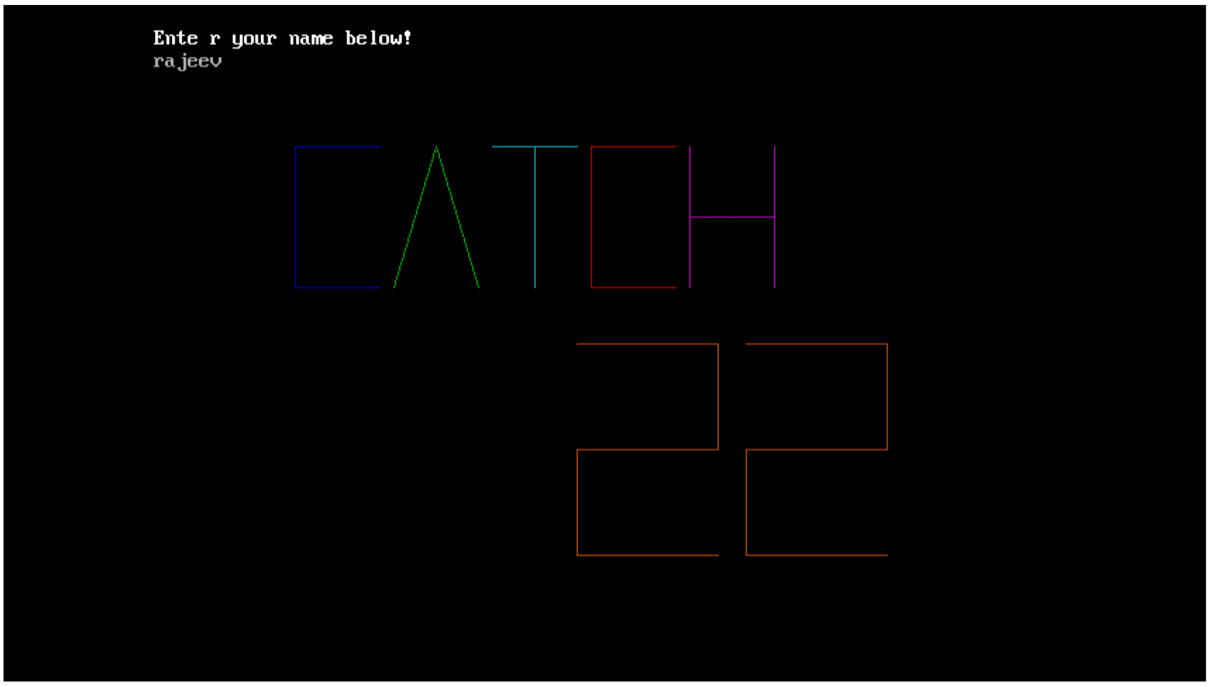




```
1.Flappy fish

2.Catch 22

3.Tic Tac Toe

4.Exit
```






```
Ente r your name below!
ra jeev
```