EXAMPLE PROGRAM SPOT THE BALL

1. Requirements

Design and implement (in Ada) a "spot the ball" computeer game. The rules for the game are as follows.

1) The "ball" is placed randomly within a 256x256 grid and its location defined by a pair of X and Y coordinate (i.e. two integers ranging from 1 to 256 inclusive).
2) The user may then enter a guess in terms of an X and Y coordinate.
3) The program should then respond with "bull's-eye" or a phrase indicating whether the ball is to the:
   - North,
   - North-east,
   - East,
   - South-east,
   - South,
   - South-west,
   - West or
   - North_west

   of the guess.
4) The user should only be allowed a maximum of 8 guesses. If the user fails to find the location of the ball within this number of guesses the program "wins"

Note: make use of the package `RANDOM`.

2. Design

A top down analysis of the problem is given in Table 1.

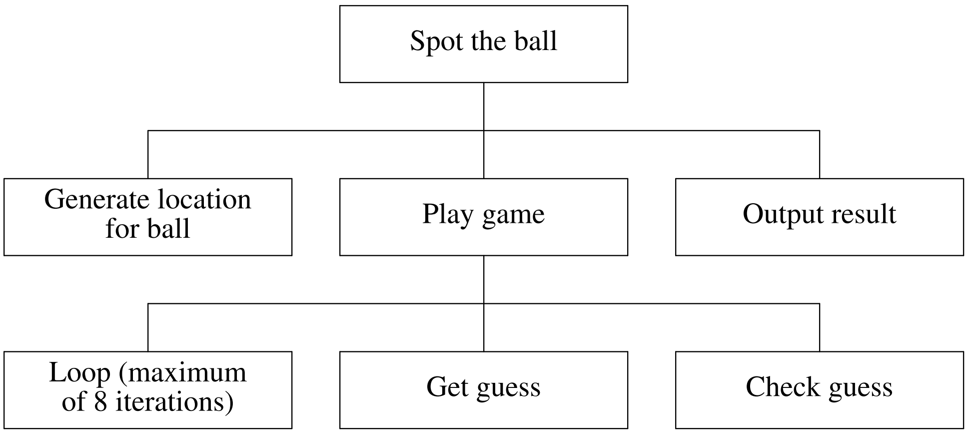


Table 1: *Top down analysis*

Where required we will define coordinates as a subtype (called COORDINATE) of the type POSITIVE with a range of 1 to 256 inclusive. We will address the identified operations using the following procedures/functions:

1) SPOT_THE_BALL (Top level procedure): Maintain "guess" loop. Calls to LOCATE_BALL and CHECK_GUESS. Output result. Include definition for subtype coordinate

Name	Description	Type	Range
X_COORD	Global input variable	COORDINATE	1..256
Y_COORD	Global input variable	COORDINATE	1..256
X_POSN	Global variable	COORDINATE	1..256
Y_POSN	Global variable	COORDINATE	1..256
MAX_NUM_GUESSES	Global constant	INTEGER	8
NUM_GUESSES	Global variable	INTEGER	Default

2) LOCATE_BALL (level 2 function): Generate and return random location for ball using RANDOM package.

Name	Description	Type	Range
X_POS	Formal (out) parameter	COORDINATE	1..256
Y_POS	Formal (out) parameter	COORDINATE	1..256

3) CHECK_GUESS (Level 2 function) : Check guess against position of ball. Return TRUE if OK, FALSE otherwise. Calls to CORRECT_X_COORD, BALL_TO_WEST and BALL_TO_EAST.

Name	Description	Type	Range
X_CRD	Formal parameter	COORDINATE	1..256
X_POS	Formal parameter	COORDINATE	1..256
Y_CRD	Formal parameter	COORDINATE	1..256
Y_POS	Formal parameter	COORDINATE	1..256

4) CORRECT_X_COORD (level 3 function): Check whether Y guess is les than, equal to or greater than Y coordinate of ball where the X coordinate of the ball has been correctly guessed. Return TRUE if equal, and FALSE otherwise.

Name	Description	Type	Range
Y_CRD	Formal parameter	COORDINATE	1..256
Y_POS	Formal parameter	COORDINATE	1..256

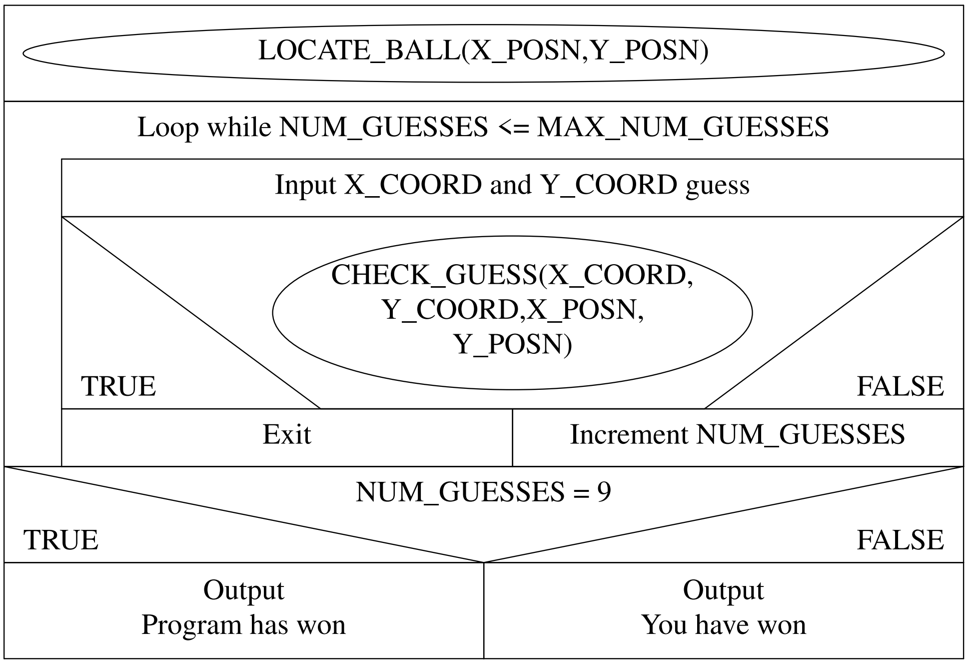
5) `BALL_TO_WEST` (level 3 procedure): Check whether Y guess is les than, equal to or greater than Y coordinate of ball where it is known that the ball is to the west of the current guess.

Name	Description	Type	Range
Y_CRD	Formal parameter	COORDINATE	1..256
Y_POS	Formal parameter	COORDINATE	1..256

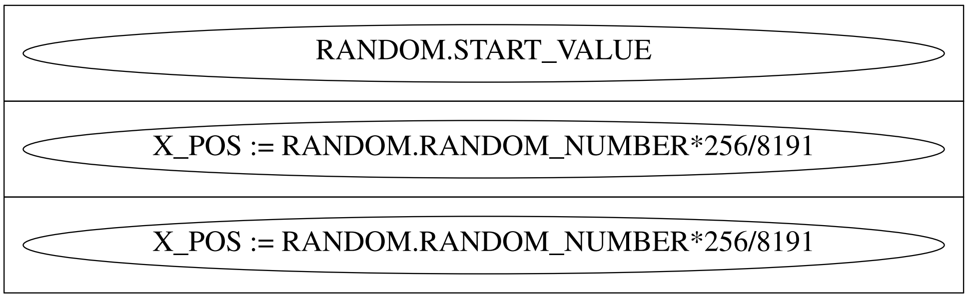
6) `BALL_TO_EAST` (level 3 procedure): Check whether Y guess is les than, equal to or greater than Y coordinate of ball where it is known that the ball is to the east of the current guess.

Name	Description	Type	Range
Y_CRD	Formal parameter	COORDINATE	1..256
Y_POS	Formal parameter	COORDINATE	1..256

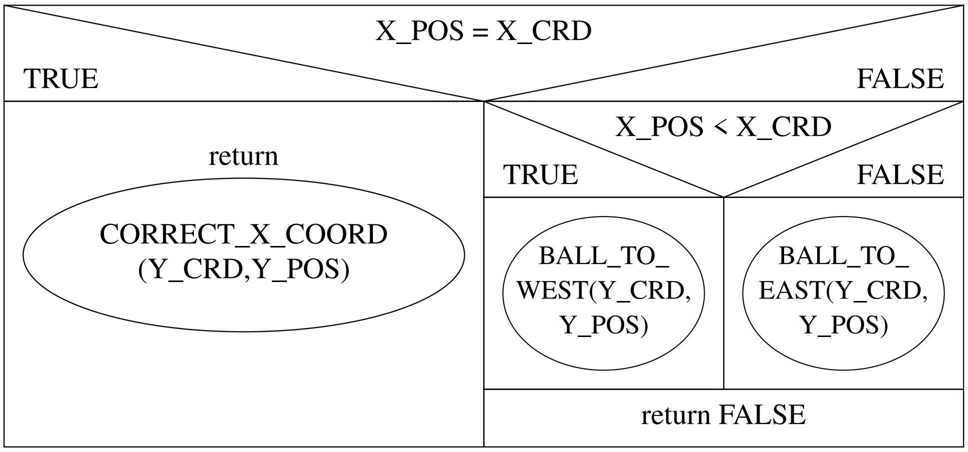
A complete Nassi-Shneiderman design for the above is given in Table 2.



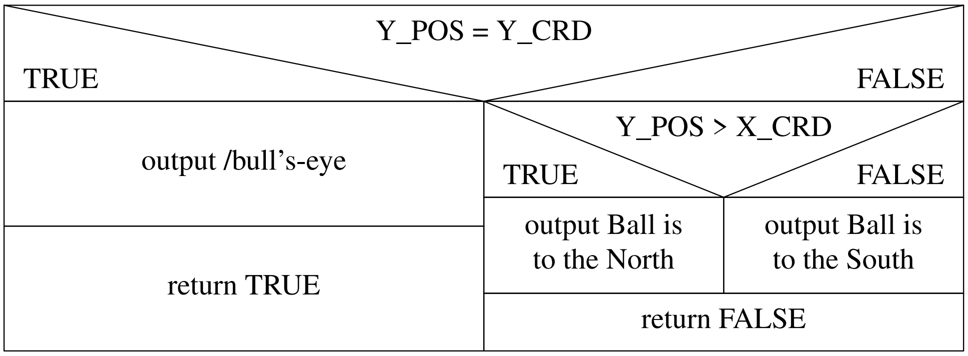
SPOT_THE_BALL
(Top level procedure)



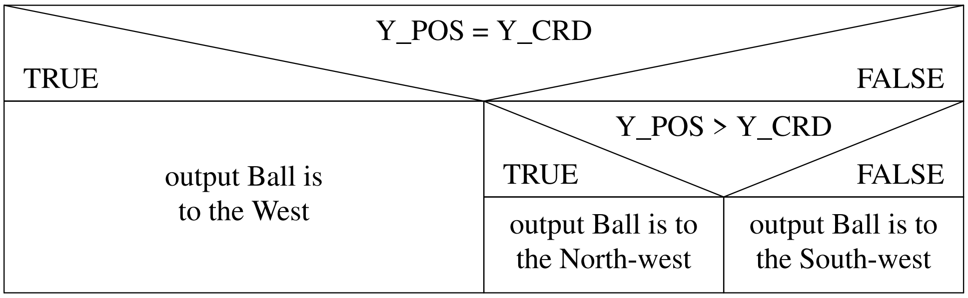
LOCATE_BALL
(Level 2 procedure)



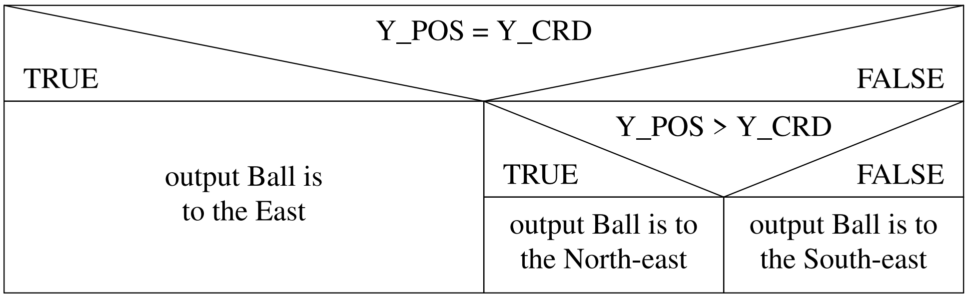
CHECK_GUESS
(Level 2 function)



CORRECT_X_COORD
(Level 3 function)



BALL_TO_WEST
(Level 3 procedure)



BALL_TO_EAST
(Level 3 procedure)

Table 2*: Nassi-Shneiderman chart*

A flow chart indicating the flow of control through this sequence of procedures/functions is given in Tables 3(a) and (b).

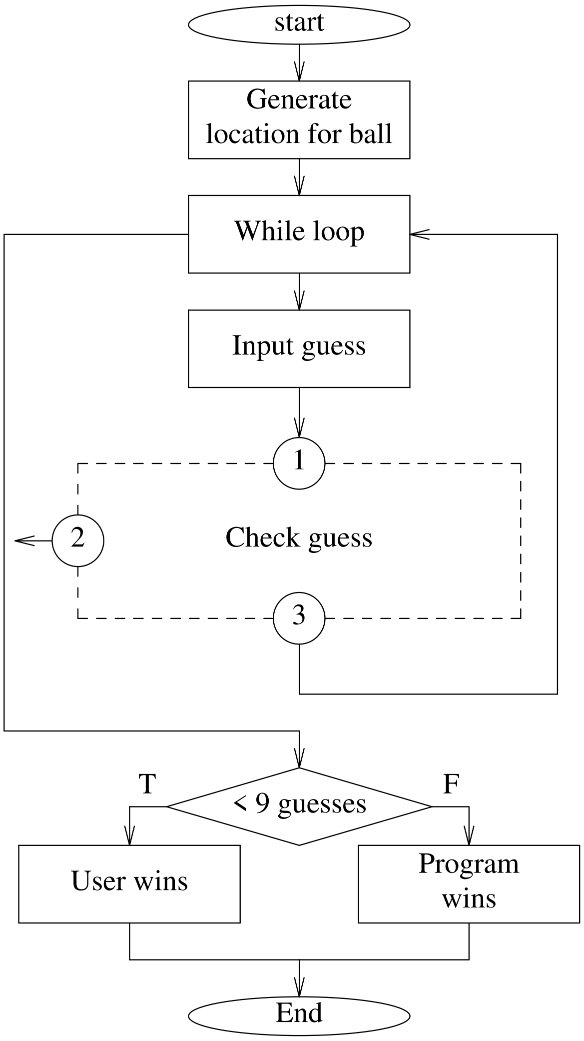


Table 3a*: Top level Control Flow Chart*

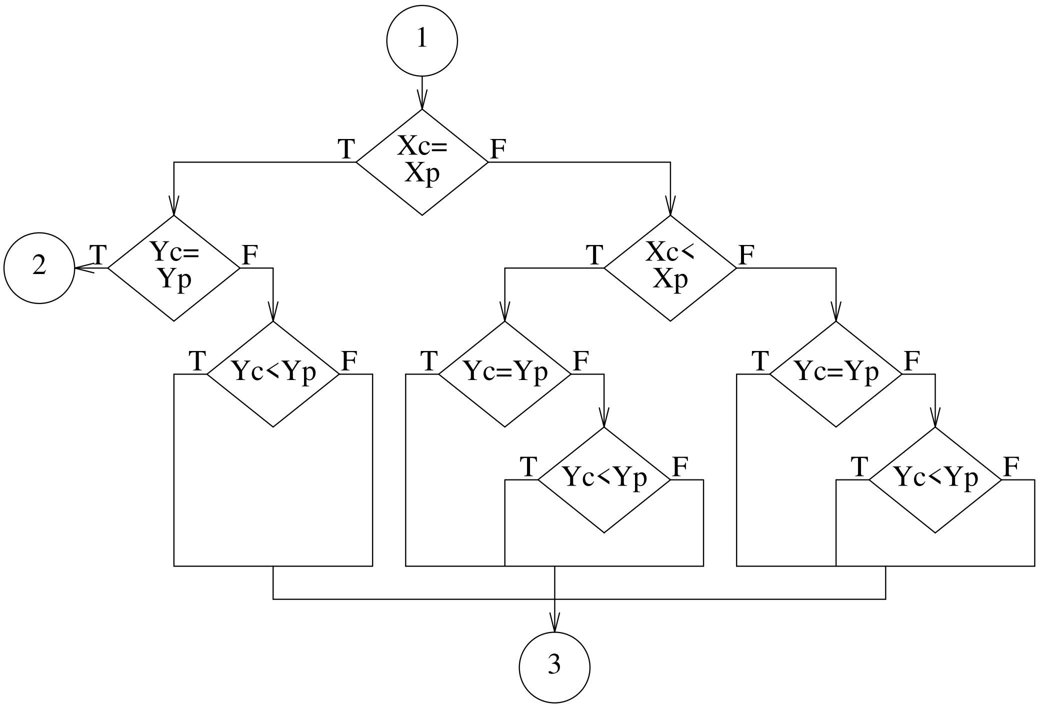


Table 3b: *Control Flow Chart for Check Guess*

3. Implementation

The encoding of the above design is given in Table 4.

```
-- SPOT THE BALL
-- 12 October 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO, RANDOM;
use TEXT_IO;

procedure SPOT_THE_BALL is
    package INTEGER_INOUT is new INTEGER_IO(INTEGER);
    use INTEGER_INOUT;
    subtype COORDINATE is POSITIVE range 1..256;
    X_COORD, Y_COORD : COORDINATE;
    X_POSN, Y_POSN   : COORDINATE;
    MAX_NUM_GUESSES  : constant := 8 ;
    NUM_GUESSES      : INTEGER  := 1;
```

Table 4: *Implementation*

```
-----------------------------------------------------------------
-- Random location of ball within space
procedure LOCATE_BALL(X_POS, Y_POS : out COORDINATE) is
begin
    RANDOM.START_VALUE;
    X_POS := RANDOM.RANDOM_NUMBER*256/8191;
    Y_POS := RANDOM.RANDOM_NUMBER*256/8191;
end LOCATE_BALL;
-----------------------------------------------------------------

-----------------------------------------------------------------
-- Check guess
function CHECK_GUESS(X_CRD, Y_CRD, X_POS, Y_POS : COORDINATE) return
        BOOLEAN is

    ---------------------------------------------------------
    -- Correct X
    function CORRECT_X_COORD(Y_CRD, Y_POS : COORDINATE) return BOOLEAN is
    begin
        if Y_POS = Y_CRD then
            PUT_LINE("bull's-eye!");
            RETURN(TRUE);
        else
            if Y_POS > Y_CRD then
                PUT_LINE("Ball is to the North");
            else
                PUT_LINE("Ball is to the South");
            end if;
            RETURN(FALSE);
        end if;
    end CORRECT_X_COORD;
    ---------------------------------------------------------
    -- Ball to west
    procedure BALL_TO_WEST(Y_CRD, Y_POS : COORDINATE) is
    begin
        if Y_POS = Y_CRD then
            PUT_LINE("Ball is to the West");
        else
            if Y_POS > Y_CRD then
                PUT_LINE("Ball is to the North-west");
            else
                PUT_LINE("Ball is to the South-west");
            end if;
        end if;
    end BALL_TO_WEST;
    ---------------------------------------------------------
```

Table 4 Continued

```
    -- Ball to east
    procedure BALL_TO_EAST(Y_CRD, Y_POS : COORDINATE) is
    begin
      if Y_POS = Y_CRD then
        PUT_LINE("Ball is to the East");
      else
        if Y_POS > Y_CRD then
          PUT_LINE("Ball is to the North-east");
        else
          PUT_LINE("Ball is to the South-east");
        end if;
      end if;
    end BALL_TO_EAST;

    --------------------------------------------------------

  begin
    if X_POS = X_CRD then
      RETURN(CORRECT_X_COORD(Y_CRD,Y_POS));
    else
      if X_POS < X_CRD then
        BALL_TO_WEST(Y_CRD,Y_POS);
      else
        BALL_TO_EAST(Y_CRD,Y_POS);
      end if;
      RETURN(FALSE);
    end if;
  end CHECK_GUESS;
  ----------------------------------------------------------------

begin
-- Locater ball
  LOCATE_BALL(X_POSN,Y_POSN);
-- Loop through maximum of 8 guesses
  while NUM_GUESSES <= MAX_NUM_GUESSES loop
    PUT_LINE("Input your guess (X and Y coordinates between 1 and 256): ");
    GET(X_COORD);
    GET(Y_COORD);
    exit when (CHECK_GUESS(X_COORD,Y_COORD,X_POSN,Y_POSN));
    NUM_GUESSES := NUM_GUESSES+1;
  end loop;
-- Final output
  if NUM_GUESSES = 9 then
    PUT_LINE("Program has won");
  else
    PUT_LINE("You have won");
  end if;
end SPOT_THE_BALL;
```

Table 4 Continued

4. Testing

First note that we do not need to test the package - it is assumed that this has been designed, implemented and tested separately by the developer and that consequently it is fully operational.

4.1. Black box testing

BVA and limit Testing: The input coordinates should be tested as shown in Table 5 using BVA and limit testing techniques. Results are presented in table 6.</P>

TEST CASE		RESULT
X_COORD	Y_COORD	
0	*	CONSTRAINT_ERROR
1	0	CONSTRAINT_ERROR
2	1	OK
2	2	OK
255	255	OK
255	256	OK
256	257	CONSTRAINT_ERROR
257	*	CONSTRAINT_ERROR

Table 5: Input Test Cases

```
kuban-206 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
5001
Input your guess (X and Y coordinates between 1 and 256):
0
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week11/spotTheBall.ada
on line 101. Rangecheck error: 0 not in range 1..256.

kuban-208 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
477
Input your guess (X and Y coordinates between 1 and 256):
1 0
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week11/spotTheBall.ada
on line 102. Rangecheck error: 0 not in range 1..256.
```

Table 6: Black box testing result

```
kuban-209 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
789
Input your guess (X and Y coordinates between 1 and 256):
2 1
Ball is to the North-east
Input your guess (X and Y coordinates between 1 and 256):
2 2
Ball is to the North-east
Input your guess (X and Y coordinates between 1 and 256):
255 255
Ball is to the South-west
Input your guess (X and Y coordinates between 1 and 256):
255 256
Ball is to the South-west
Input your guess (X and Y coordinates between 1 and 256):
256 257
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week11/spotTheBall.ada
on line 102. Rangecheck error: 257 not in range 1..256.

kuban-210 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
199
Input your guess (X and Y coordinates between 1 and 256):
257
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week11/spotTheBall.ada
on line 101. Rangecheck error: 257 not in range 1..256.
```

Table 6 continued

4.2. White box testing

Path Testing: The DFD given above identifies the paths through the system. Test cases should be generated to ensure that every path is exercised. A seed of `123` wile produce `X_POS = 235` and `Y_POS = 109`. Using this seed the sequence of "guesses" presented in Table 7 will test all the paths in code which may influence the "advice". Note that after 8 iterations the loop is complete (program has won). The final path, where a correct answer is guessed is exercised bo the loop test cases considered next. (Results presented in Table 8.)

"GUESSES"		RESULT
X_COORD	Y_COORD	
250	250	Ball is to the South-west
250	109	Ball is to the West
250	10	Ball is to the North-west
235	250	Ball is to the South
235	10	Ball is to the North
10	250	Ball is to the South-east
10	109	Ball is to the East
10	10	Ball is to the North-east Program has won

Table 5: Path Test Cases

```
kuban-220 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
123
Input your guess (X and Y coordinates between 1 and 256):
250 250
Ball is to the South-west
Input your guess (X and Y coordinates between 1 and 256):
250 109
Ball is to the West
Input your guess (X and Y coordinates between 1 and 256):
250 10
Ball is to the North-west
Input your guess (X and Y coordinates between 1 and 256):
235 250
Ball is to the South
Input your guess (X and Y coordinates between 1 and 256):
235 10
Ball is to the North
Input your guess (X and Y coordinates between 1 and 256):
10 250
Ball is to the South-east
Input your guess (X and Y coordinates between 1 and 256):
10 109
Ball is to the East
Input your guess (X and Y coordinates between 1 and 256):
10 10
Ball is to the North-east
Program has won
```

Table 8: Path testing result

Loop Testing: we should also test that the loop is working appropriately. In this case we should test the loop at 1, 2 4 7 and 8 iterations (the last has already been taken into account as part of the proposed path test cases). Using the same seed and the inputs given above, and by imposing a correct guess where appropriate, we cause the loop to repeat 1, 2, 4 and 7 (Table 9, results in Table 10).

"GUESSES"		RESULT
X_COORD	Y_COORD	
235	109	bull’s-eye! You have won

"GUESSES"		RESULT
X_COORD	Y_COORD	
250 235	250 109	Ball is to the South-west bull’s-eye! You have won

"GUESSES" X_COORD	Y_COORD	RESULT
250	250	Ball is to the South-west
250	109	Ball is to the West
250	10	Ball is to the North-west
235	109	bull's-eye! You have won

"GUESSES" X_COORD	Y_COORD	RESULT
250	250	Ball is to the South-west
250	109	Ball is to the West
250	10	Ball is to the North-west
235	250	Ball is to the South
235	10	Ball is to the North
10	250	Ball is to the South-east
10	109	Ball is to the East
235	109	bull's-eye! You have won

Table 9: Loop Test Cases

```
Enter seed (odd number within the range 1 to  8191):
123
Input your guess (X and Y coordinates between 1 and 256):
235 109
bull's-eye!
You have won

kuban-222 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
123
Input your guess (X and Y coordinates between 1 and 256):
250 250
Ball is to the South-west
Input your guess (X and Y coordinates between 1 and 256):
235 109
bull's-eye!
You have won
```

Table 10: Loop testing result

```
kuban-223 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
123
Input your guess (X and Y coordinates between 1 and 256):
250 250
Ball is to the South-west
Input your guess (X and Y coordinates between 1 and 256):
250 109
Ball is to the West
Input your guess (X and Y coordinates between 1 and 256):
250 10
Ball is to the North-west
Input your guess (X and Y coordinates between 1 and 256):
235 109
bull's-eye!
You have won

kuban-224 $ spot_the_ball
Enter seed (odd number within the range 1 to  8191):
123
Input your guess (X and Y coordinates between 1 and 256):
250 250
Ball is to the South-west
Input your guess (X and Y coordinates between 1 and 256):
250 109
Ball is to the West
Input your guess (X and Y coordinates between 1 and 256):
250 10
Ball is to the North-west
Input your guess (X and Y coordinates between 1 and 256):
235 250
Ball is to the South
Input your guess (X and Y coordinates between 1 and 256):
235 10
Ball is to the North
Input your guess (X and Y coordinates between 1 and 256):
10 250
Ball is to the South-east
Input your guess (X and Y coordinates between 1 and 256):
10 109
Ball is to the East
Input your guess (X and Y coordinates between 1 and 256):
235 109
bull's-eye!
You have won
```

Table 10 continued