PROJECT 2

Craps game

CIS-5 46740
Chris Ramos
08/01/2020

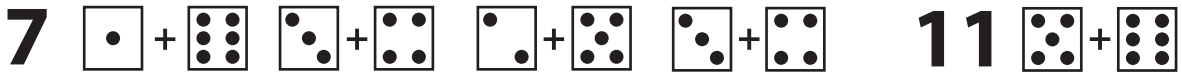
Introduction

In the second version of this craps game, certain parts of the code is broken up at appropriate places and is inserted into functions to make the code more organized and readable. Using a function for the rolling of the dice makes it easily reusable in other parts of the code by using a simple function call.

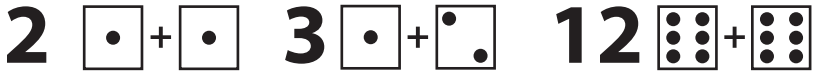
Rules of craps

On the first roll

If the player rolls a 7 or 11 they **win**.



If the player rolls a 2, 3 or 4 they **lose**.



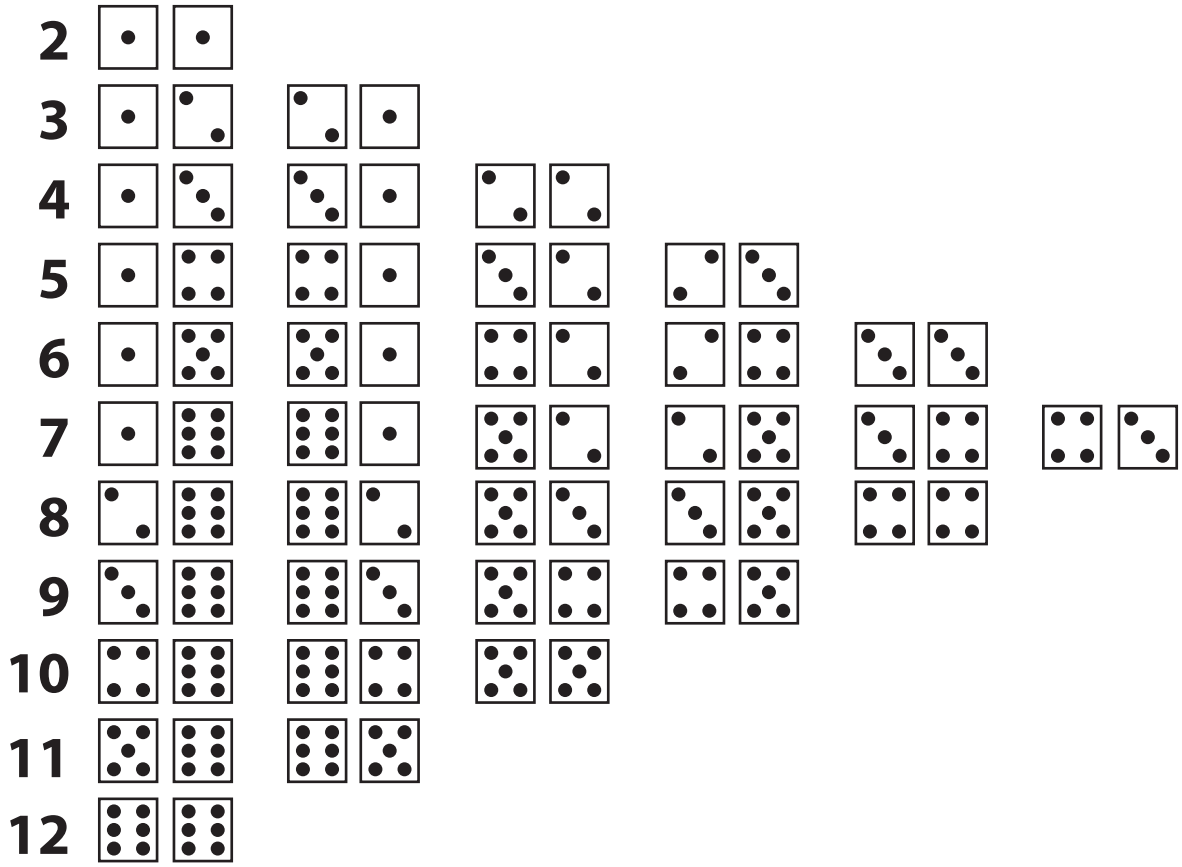
If the player rolls a **4, 5, 6, 8, 9 or 10** their point is set and they **Roll again.**

Example:

If the player gets a 4 on the first roll their point is 4 and they roll again.

On the second roll they will have to get another 4 before a 7 to win.

All possible combinations for each number



Functions added

showRules() function

Displays the rules of the game with a header pulled from an external file. The text is formated using setw() to add space and centering making it more easily readable.

```
                RULES OF CRAPS
************************************************
             On the first roll
7 or 11                              You win
2, 3, 12                            You lose
4, 5, 6, 8, 9, 10                 Roll again

            On the second roll
    If you rolled a number from the roll
  again row get it again before a 7 to win.
```

getBet() function

Displays how much the house pays and what pecentage it gets. It then prompts the player to enter their bet and stores it in the variable named **bet**.

```
The house pays double if you win
but takes a 2% cut.
Place your bet: $500
```

startGame() function

Prompts the player to enter the character S to start the game. Their choice is stored in a variable named **start.** If they enter any other character a while loop is initilized and will keep looping until an S in entered.

```
Enter a S to start the game: t

You must enter an S to start: y

You must enter an S to start: g

You must enter an S to start: s
```

rollDice() function

Uses the random number generator to store a random number in the variables named **dice1** and **dice2**. The sum of the two variables is stored and returned in a varible called **sum** which determines the result of the first roll.

If the player doesnt win or lose on the first roll, and is required to roll again the function is continuously called till the player gets their point and wins or they roll a 7 before their point and losses.

```
you rolled a 5 and 3 = 8

your point is 8
```

getResults() function

A switch statement takes the return value of the **sum** variable from the **rollDice** function and determines the result of the first roll. A string variable named **result** holds the value.

If the roll is a	
2, 3, or 12	result = lose
7 or 11	result = win
4, 5, 6, 8, 9, 10	result = point

If **result** is point, point is set to equal **sum** and the player will have to roll that number before they roll a 7 to win.

A variable named **rollNum** increments each roll and is included in each case to count the first roll.

```
you rolled a 5 and 2 = 7


CONGRATULATIONS! YOU WIN!
You won in 1 roll(s)
```

```
you rolled a 1 and 2 = 3


CRAPS! YOU LOSE
You lost in 1 roll(s)
```

```
you rolled a 2 and 4 = 6

your point is 6
```

```
        WINNINGS
++++++++++++++++++++++++
You bet: $1000.00
You won: $2000.00
House gets: $40.00
You take home: $1960.00
```

CalcDispWin() function

If the player wins, the amount they entered in the **getBet()** function is used to calculate their winnings.

To determine how much the player wins, a variable named **usrWins** multiplies the bet by two, because the house pays double the bet.

To determine how much the house gets, a variable named **houWins** takes the 2% multiples it by 100 to convert it to decimal than multiplies it by the players winnings that stored in the **usrWins** variable.

To determine how much the player takes home, a variable named **totUwns** subtracts the result of **usrWins** from the result of **houWins**

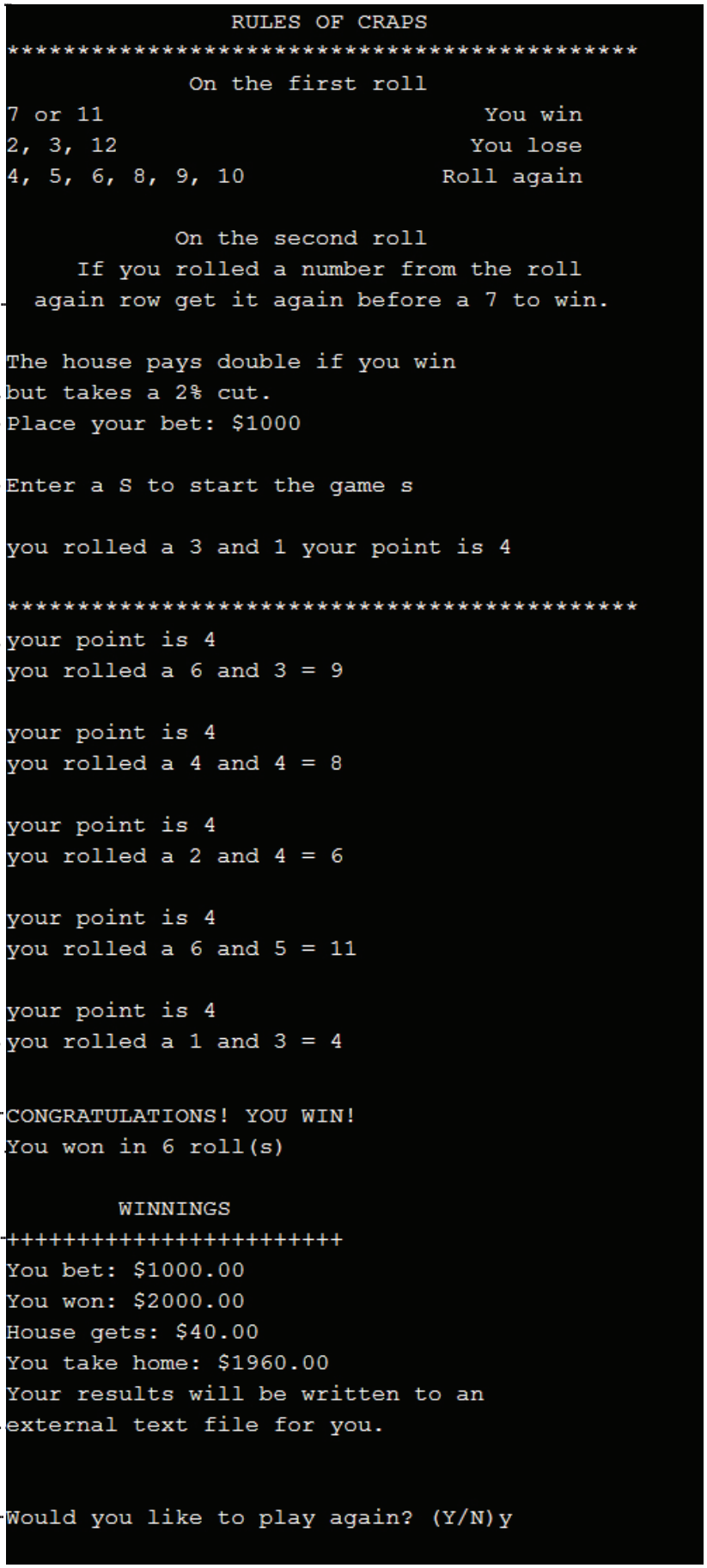
```
        LOSSES
------------------
You bet: $1000
You lost: $1000
```

DisplayLosses() function

If the player loses the amount they entered in the **getBet()** function is displayed along with how much they lost.

Inputs and outputs

Project size: 334 lines
Number of variables: 16
Versions: 4



Rules displayed for player

Terms of bet displayed

Input for player to place their bet

Input for player to the game by entering an S

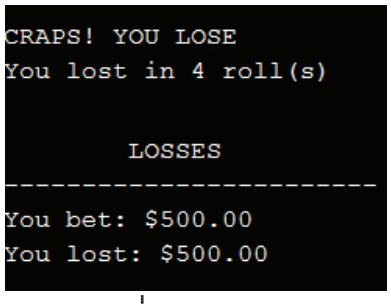
1st roll initialized

If there is no win or loss on the first roll the player rolls again and keeps rolling till they get their point and win or they roll a 7 and lose.

Congratulations message displayed if the player wins along with how many rolls they won in.

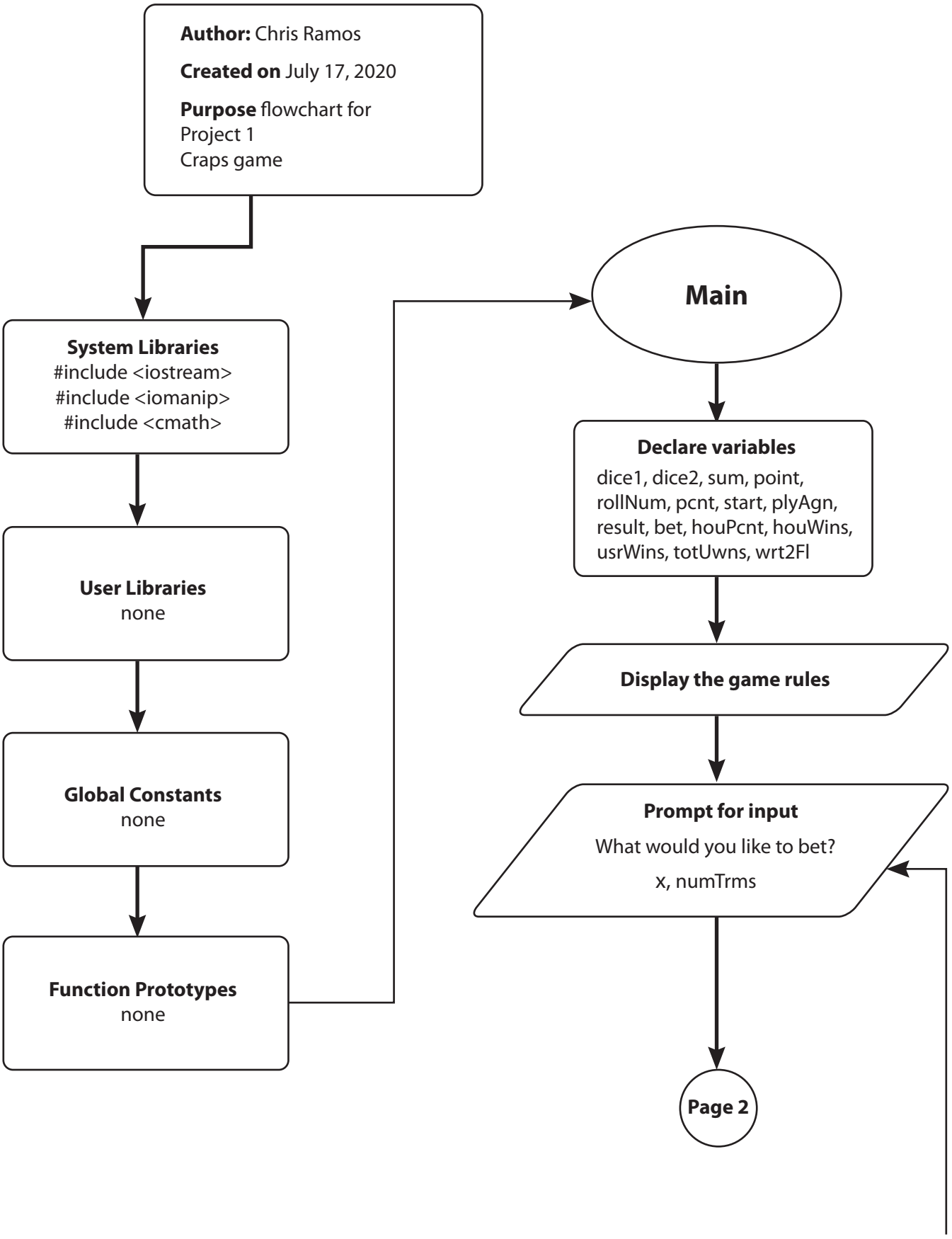
Winnings displayed and along with message infroming user that the results have been written to an external file.

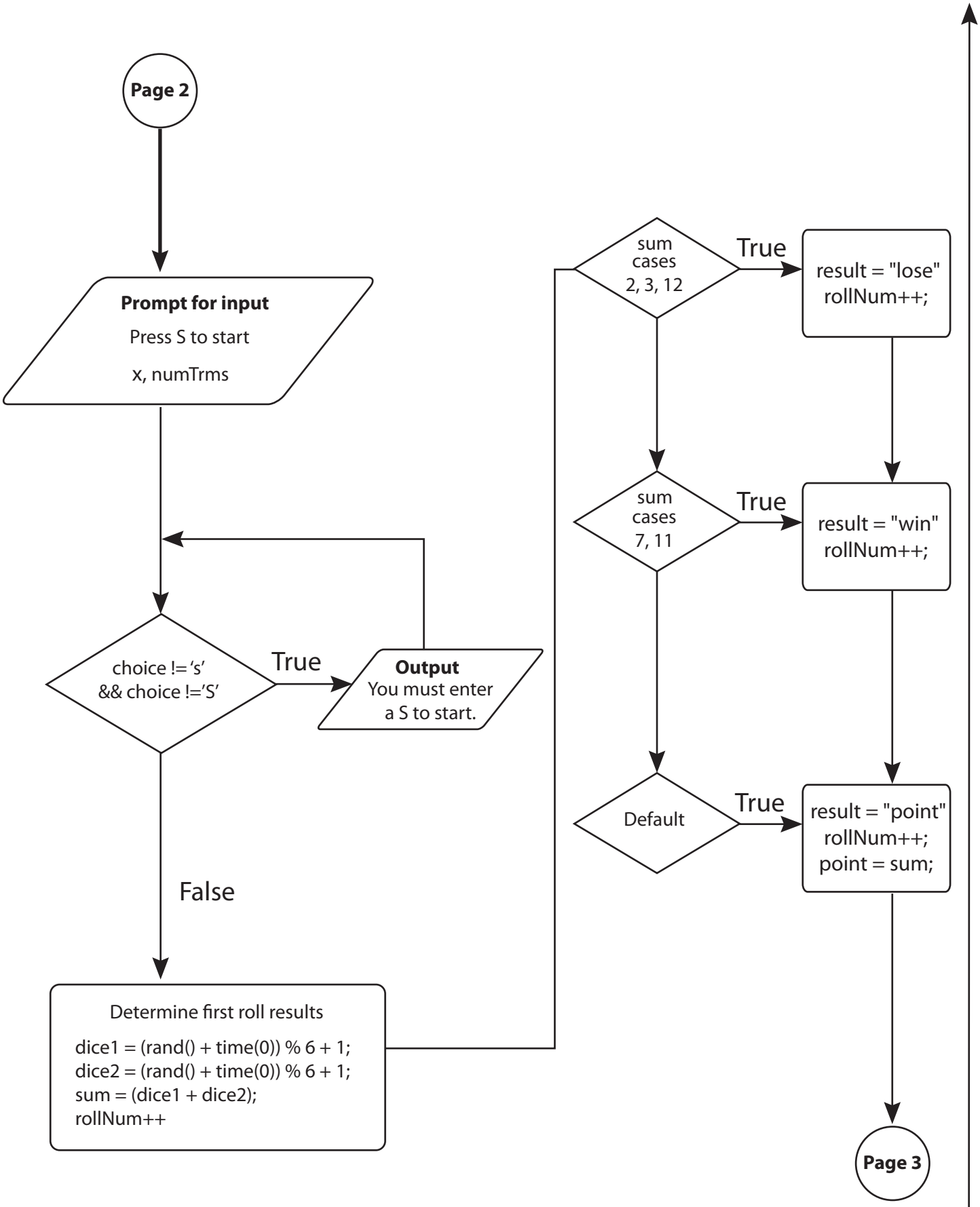
Input for player to choose the option to play again.

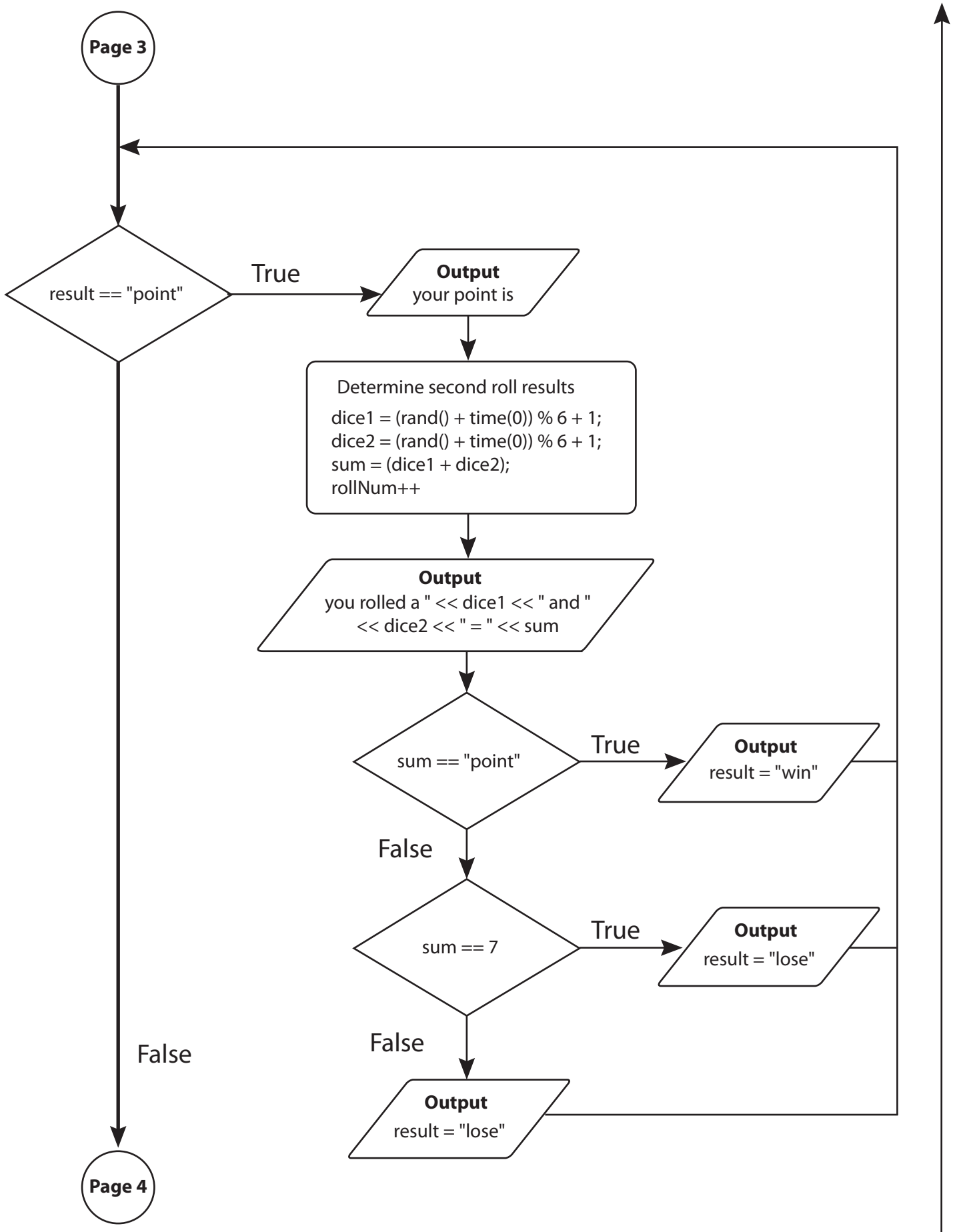


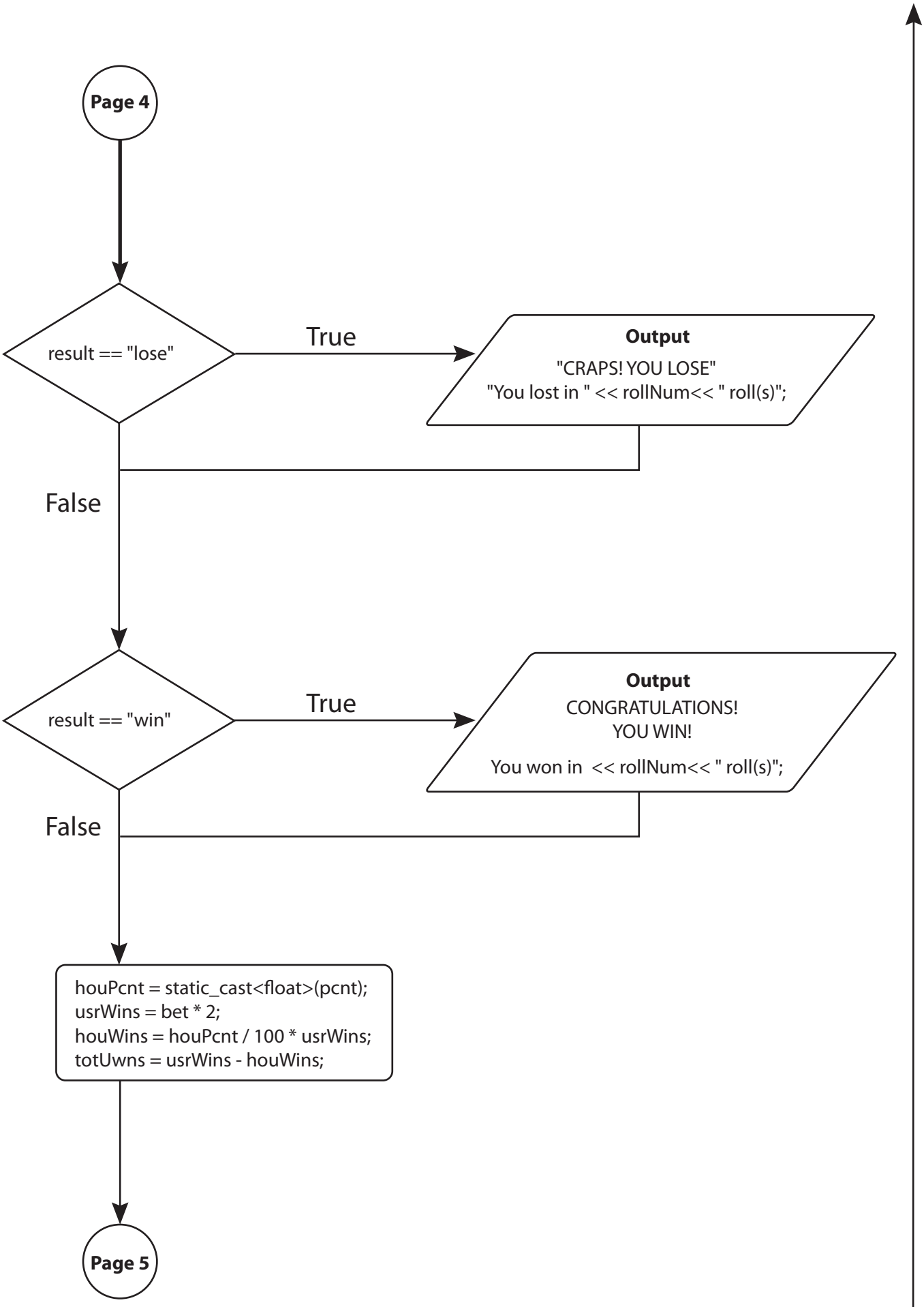
If the player loses a message is displayed showing how many rolls it took and their losses.

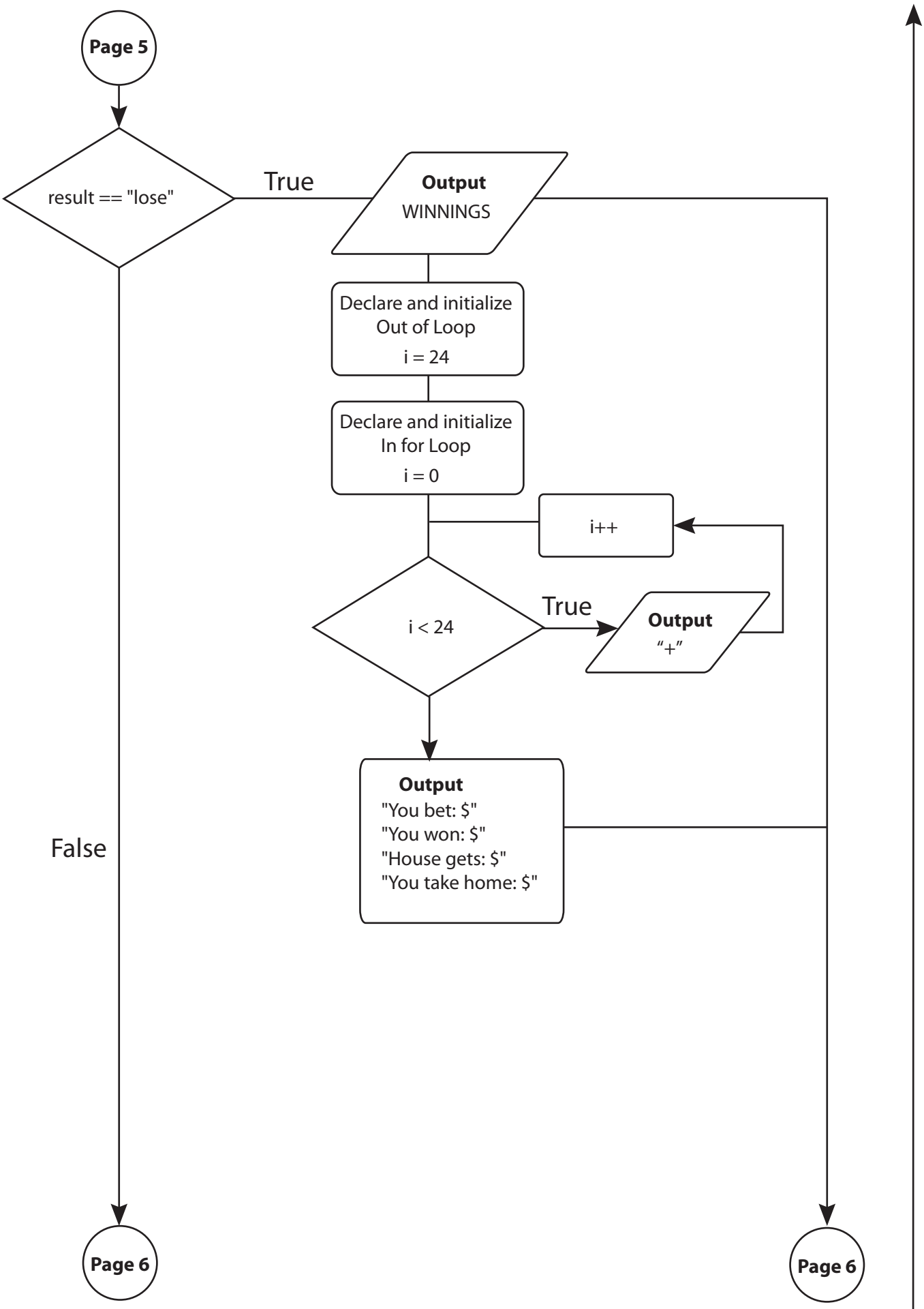
Flow chart

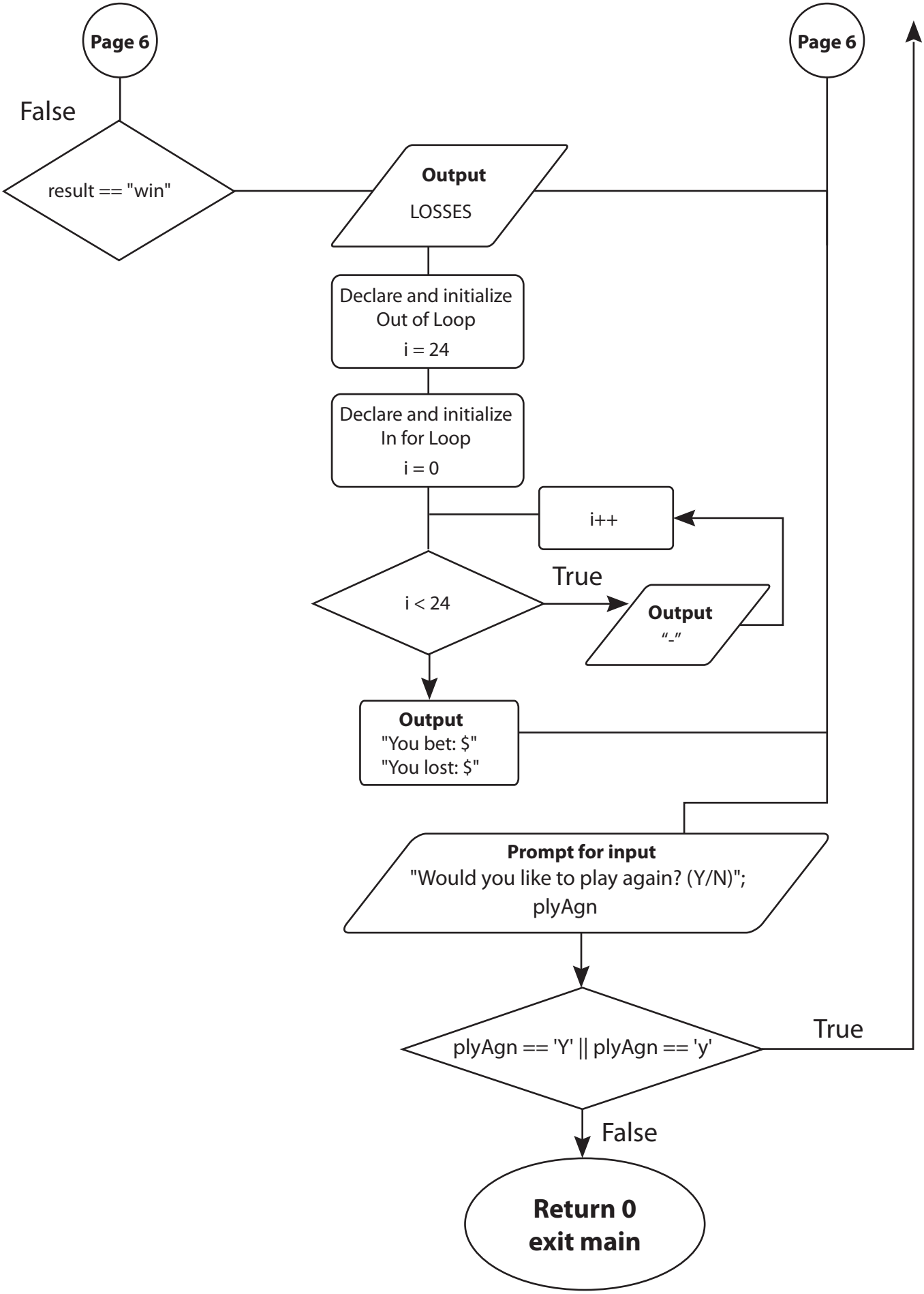

Author: Chris Ramos
Created on July 17, 2020
Purpose flowchart for Project 1 Craps game
System Libraries
#include <iostream>
#include <iomanip>
#include <cmath>
User Libraries
none
Global Constants
none
Function Prototypes
none
Main
Declare variables
dice1, dice2, sum, point, rollNum, pcnt, start, plyAgn, result, bet, houPcnt, houWins, usrWins, totUwns, wrt2Fl
Display the game rules
Prompt for input
What would you like to bet?
x, numTrms
Page 2


Page 2
Prompt for input
Press S to start
x, numTrms
choice != 's'
&& choice !='S'
True
Output
You must enter
a S to start.
False
Determine first roll results
dice1 = (rand() + time(0)) % 6 + 1;
dice2 = (rand() + time(0)) % 6 + 1;
sum = (dice1 + dice2);
rollNum++
sum
cases
2, 3, 12
True
result = "lose"
rollNum++;
sum
cases
7, 11
True
result = "win"
rollNum++;
Default
True
result = "point"
rollNum++;
point = sum;
Page 3


Page 3
result == "point"
True
Output
your point is
Determine second roll results
dice1 = (rand() + time(0)) % 6 + 1;
dice2 = (rand() + time(0)) % 6 + 1;
sum = (dice1 + dice2);
rollNum++
Output
you rolled a " << dice1 << " and "
<< dice2 << " = " << sum
sum == "point"
True
Output
result = "win"
False
sum == 7
True
Output
result = "lose"
False
Output
result = "lose"
False
Page 4


Page 4
result == "lose"
True
Output
"CRAPS! YOU LOSE"
"You lost in " << rollNum<< " roll(s)";
False
result == "win"
True
Output
CONGRATULATIONS!
YOU WIN!
You won in << rollNum<< " roll(s)";
False
houPcnt = static_cast<float>(pcnt);
usrWins = bet * 2;
houWins = houPcnt / 100 * usrWins;
totUwns = usrWins - houWins;
Page 5


Page 5
result == "lose"
True
Output
WINNINGS
Declare and initialize
Out of Loop
i = 24
Declare and initialize
In for Loop
i = 0
i++
i < 24
True
Output
"+"
Output
"You bet: $"
"You won: $"
"House gets: $"
"You take home: $"
False
Page 6
Page 6


Page 6
Page 6
False
result == "win"
Output
LOSSES
Declare and initialize
Out of Loop
i = 24
Declare and initialize
In for Loop
i = 0
i++
True
i < 24
Output
"-"
Output
"You bet: $"
"You lost: $"
Prompt for input
"Would you like to play again? (Y/N)";
plyAgn
plyAgn == 'Y' || plyAgn == 'y'
True
False
Return 0
exit main

Pseudo code

• Display the game rules.

• Display betting rules.

• Prompt the player to place their bet

• Prompt the player to start the game.

• Validate the users choice.

• Execute the first roll.

• Display the results of the first roll.

• If they roll a 2, 3, or 12 tell the player that they losts and display what they bet and what they lost.

• If they roll a 7 or 11 tell the player that they won and display what they bet, how much they won, how much the house gets and how much they get to take home.

• If they roll anything else (4, 5, 6, 8, 9 or 10) set that number as their point and display a message that shows them the number.

• Once the point is set execute the second roll.

• Display the results of the second roll that includes the players point and what the the number of each dice is and the sum of the two numbers.

• If the player rolls a 7 before their point tell them that they lost and display how many rolls it took them to lose, what they bet and what they lost.

• If the player rolls their point before a 7 tell them that they won and display how many rolls it took them to win, what they bet, how much they won, how much the house gets and how much they get to take home.

• If the player rolls any number that isnt a 7 or isnt their point display their point and what the the number of each dice is and the sum of the two numbers and roll again and kepp rolling till they win or lose.

• After the player wins or loses aske them if they would like to play again.

Versions

Version 1

Game rules are embedded into a function named **showRules()** that includes a headline pulled from a file, and formated text.

A function is created named **rollDice()** that includes the random number generator to simulate the roll of both dice and calculates and returns the sum.

Version 2

A function named **getBet()** is added to display the betting rules and the player is given the ability to place their bet.

A function is added named **startGame()** that prompts the player to enter an S to start the game. A while loop is used to validate the players choice and will keep prompting till an S is entered to initialize the first roll.

A function named **getResults()** is added conatining a switch statement that uses the result of the rollDice() function to determine if the status of the first roll is win, lose or roll again. It also contains a variable to include the first roll in the total count of rolls that was used to win or lose.

Version 3

A function is added named **CalcDispWin()** that uses the dollar amount entered by the player in the **getBet()** function to calculate and display how much, the user won, how much the house gets and how much the player gets to take home.

A function is added named **DisplayLosses()** that displays the players betand how much they lost.

Final version

A do while loop is added so the game can repeat as long as the player enters a "Y" or "y".

Code is added so that the players winnings that are displayed and calculated in the **CalcDispWin()** function are printed to an external file.

Limitations

Time constraints didnt allow for experimenting and adding more features.

No way to include any type of graphics.

Cant have a real interface.

Very limited interactivity and user control.

No real way to represent dice except to use numbers.