COMSC 200

Summer 2024

Programming Assignment 2

Worth 12.5 points (1.25% of your grade)

DUE: Thursday, 6/20/24 by 11:59 P.M. on Canvas

Late Pass Deadline: Sunday, 6/23/24 by 11:59 P.M. on Canvas

NOTE: Your submission for this assignment should be **one .h file**, **two .cpp files, and one .pdf file with screenshots of your five sample runs**. Your submission MUST include all four of these files.

The following naming convention should be used for naming your **class specification file**: firstname_lastname_Die.h

The following naming convention should be used for naming your **class implementation file**: firstname_lastname_Die.cpp

The following naming convention should be used for naming your **application program file**: firstname_lastname_app.cpp

The following naming convention should be used for naming your **sample runs file**: firstname_lastname_runs.pdf

For example, if your first name is “James” and your last name is “Smith”, then your four files would be named **James_Smith_Die.h**, **James_Smith_Die.cpp**, **James_Smith_app.cpp, and James_Smith_runs.pdf.**

Remember that the class member functions/constructor should be **implemented** in the Die.cpp file (class implementation file) – the Die.h file (class specification file) should only contain the member variables and the **prototypes** of the functions/constructor. All of the application logic should be in the app.cpp file (application program file).

NOTE: Make sure to use an include guard in your Die.h file.

COMMENTS – worth 1.25 points (10%) of your programming assignment grade:

Your program should have at least **ten (10)** different detailed comments explaining the different parts of your program. Each individual comment should be, at a minimum, a sentence explaining a particular part of your code. You should make each comment as detailed as necessary to fully explain your code. You should also number each of your comments (i.e., comment 1, comment 2, etc.).

Sample Runs – worth 1.25 points (10%) of your programming assignment grade:

You should submit screenshots of at least **five (5)** different sample runs of your program. You should also number each of your sample runs (i.e., sample run 1, sample run 2, etc.). Since each die roll is randomly determined you don’t have complete control over the output that you get, but you can still distinguish your ten test cases by having the user enter “N” (for no) at different points to stop the game, and by including test cases where the user keeps choosing “Y” (for yes) until they go over 21.

For your second programming assignment, you will be writing an **object-oriented C++ program** that lets the user play against the computer in a variation of the popular blackjack card game. In this variation of the game, two six-sided dice are used instead of cards. The dice are rolled, and the player tries to beat the computer's hidden total without going over 21.

- Each round of the game is performed as an iteration of a loop that repeats as long as the player agrees to roll the dice, and the player's total does not exceed 21.
- At the beginning of each round, the program will ask the user whether they want to roll the dice to accumulate points.
- During each round, the program simulates the rolling of two six-sided dice. It rolls the dice first for the computer, and then it asks the user if he or she wants to roll.
- The loop keeps a running total of both the computer and the user's points.
- The computer's total should remain hidden until the loop has finished.
- After the loop has finished, the computer's total is revealed, and the player with the most points without going over 21 wins. Neither player wins if there is a tie or if both players go over 21.

You will be implementing a **class** named Die (singular form of dice). Design the class from the following **UML diagram – your class should follow this UML diagram EXACTLY**:

Die
-sides: int // Number of sides on the die -value: int // The die's value
<<Constructor>> +Die(numSides: int) +setSides(numSides: int) +roll() +getSides(): int +getValue(): int

Here is a brief description of the constructor and member functions:

<<Constructor>> +Die(numSides: int)
The **constructor** accepts an argument for the number of sides for the die and performs an initial roll of the die.

+setSides(numSides: int)
The setSides member function accepts an argument for the number of sides for the die and performs a roll of the die.

+roll()
The roll member function simulates the rolling of the die. For example, if the number of sides for the die is 6, then the die’s value will be randomly chosen between 1 – 6.

+getSides(): int
The getSides member function returns the number of sides for the die.

+getValue(): int
The getValue member function returns the die's value.

After you create the Die class, then you can start working on the logic of the game itself. **Note:** the logic in your application program (app.cpp) should be divided up into multiple functions. In addition to the main function, think about what other functions you can include in the application program.

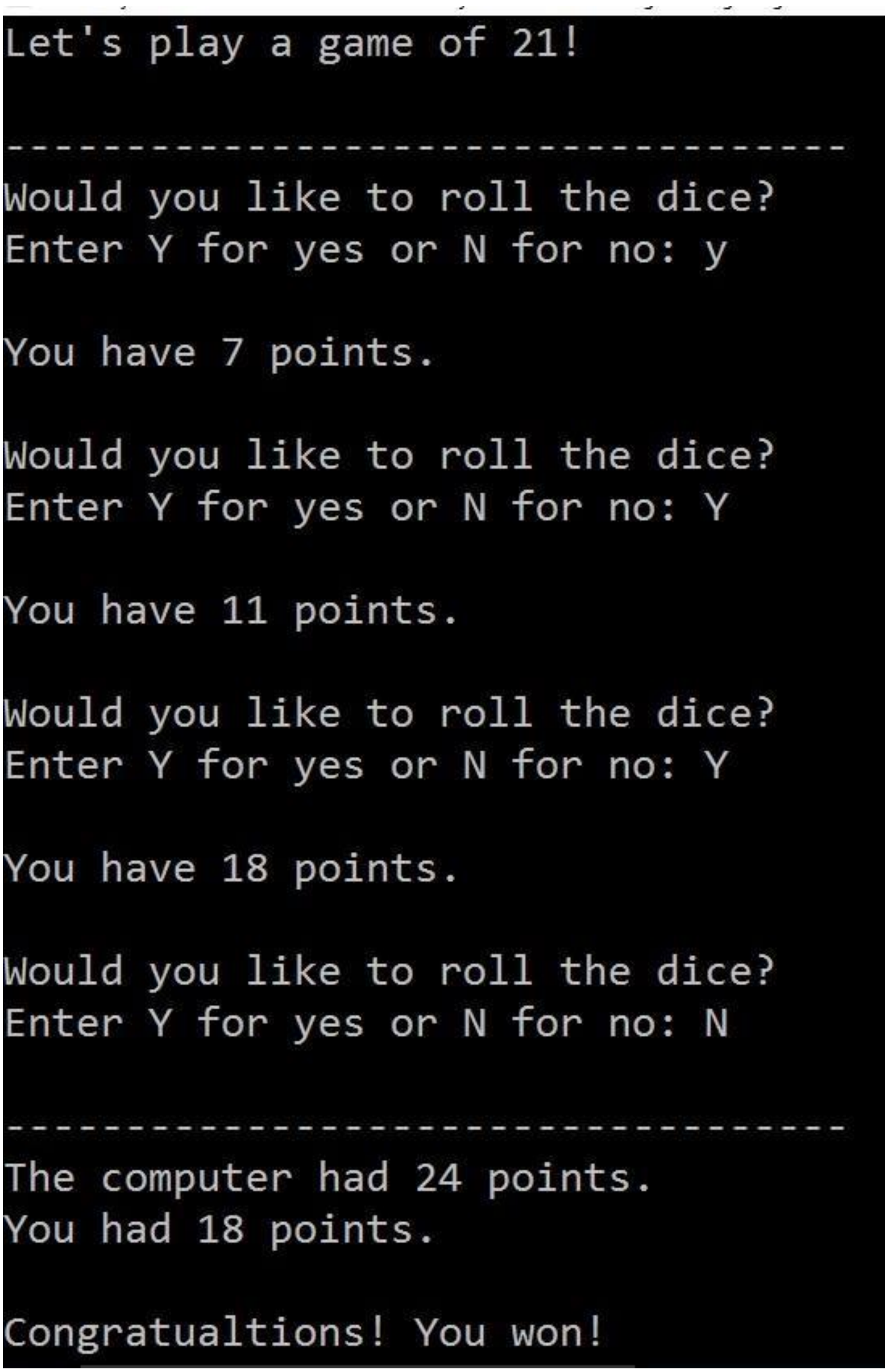
On the other hand, you’ve been given explicitly all of the functions that you need to implement in the Die class, so you do not need to come up with any additional functions in the Die class – you just need to implement everything that was listed in the **UML diagram** for the Die class.

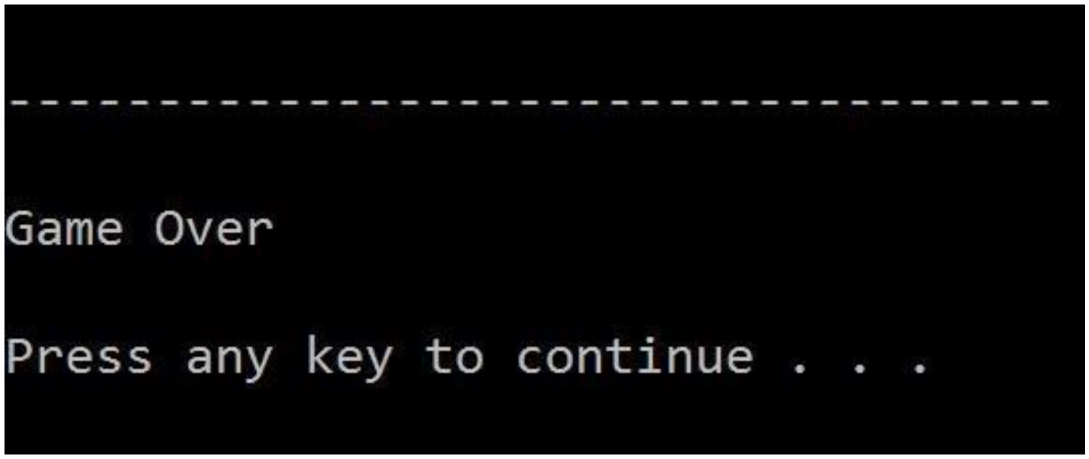
The first thing you need to do in the application program is **create two different Die objects,** because two six-sided dice are needed to play the game (the user and the computer **SHARE** the same two dice, which is why there are two die objects rather than four die objects).

NOTE: It's important to keep in mind that this Die class could be used for **other applications** besides this Blackjack game that you are implementing. For example, it could be used to implement a game that requires three dice, each with ten sides. If the Die class is implemented correctly **following the UML diagram shown above**, it could be used for any game that uses any number of dice with any number of sides. **Reusability** is very, very…… **VERY** important in object oriented programming!

The code in the **application program** (in this case a game) **uses** the Die class for the purposes of allowing the user to play the game. But the Die class is a **reusable** class that can be used for **any** game involving dice.

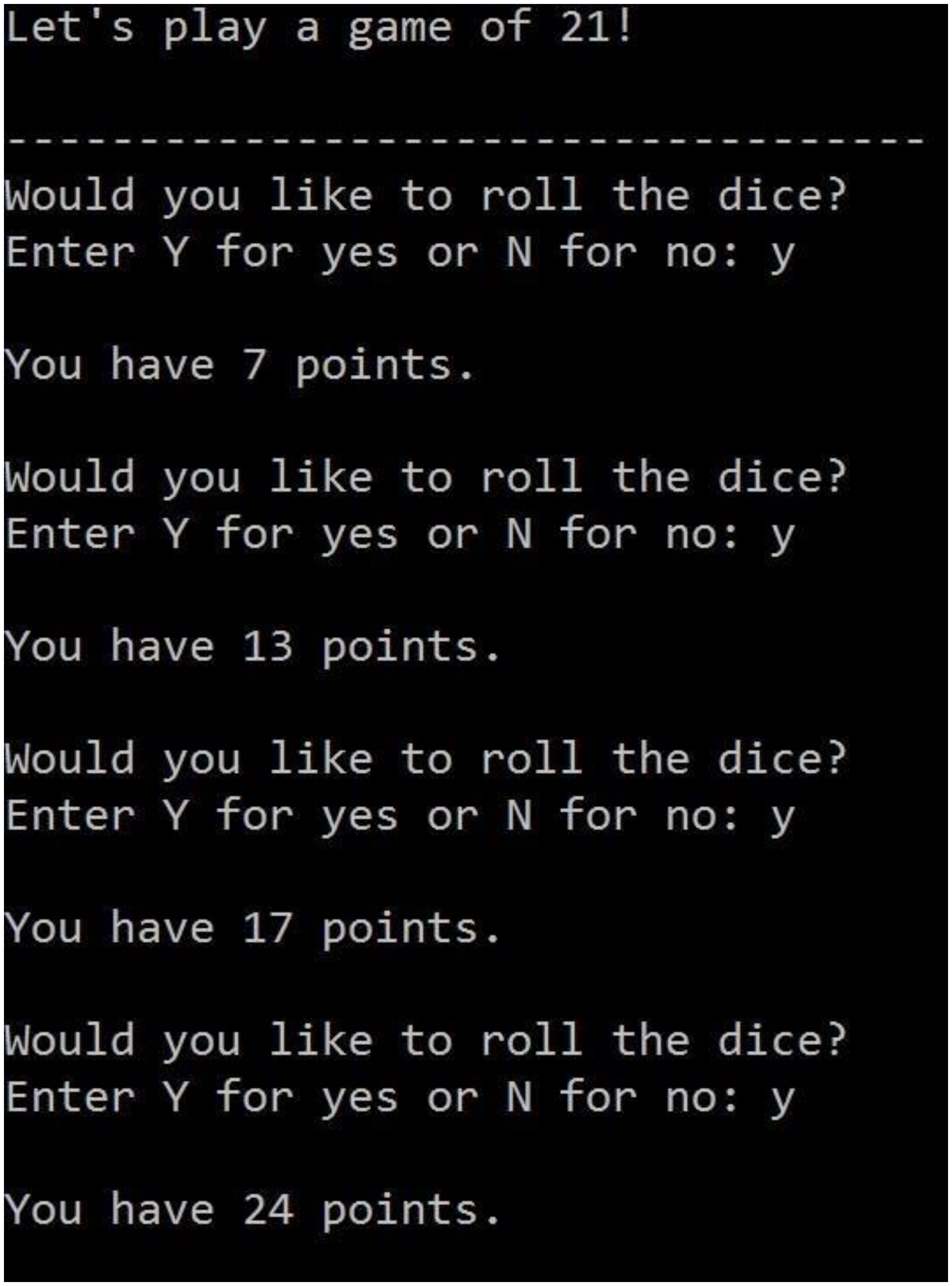
Sample Run 1:


```
Let's play a game of 21!

---------------------------------------
Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 7 points.

Would you like to roll the dice?
Enter Y for yes or N for no: Y

You have 11 points.

Would you like to roll the dice?
Enter Y for yes or N for no: Y

You have 18 points.

Would you like to roll the dice?
Enter Y for yes or N for no: N

---------------------------------------
The computer had 24 points.
You had 18 points.

Congratualtions! You won!
```

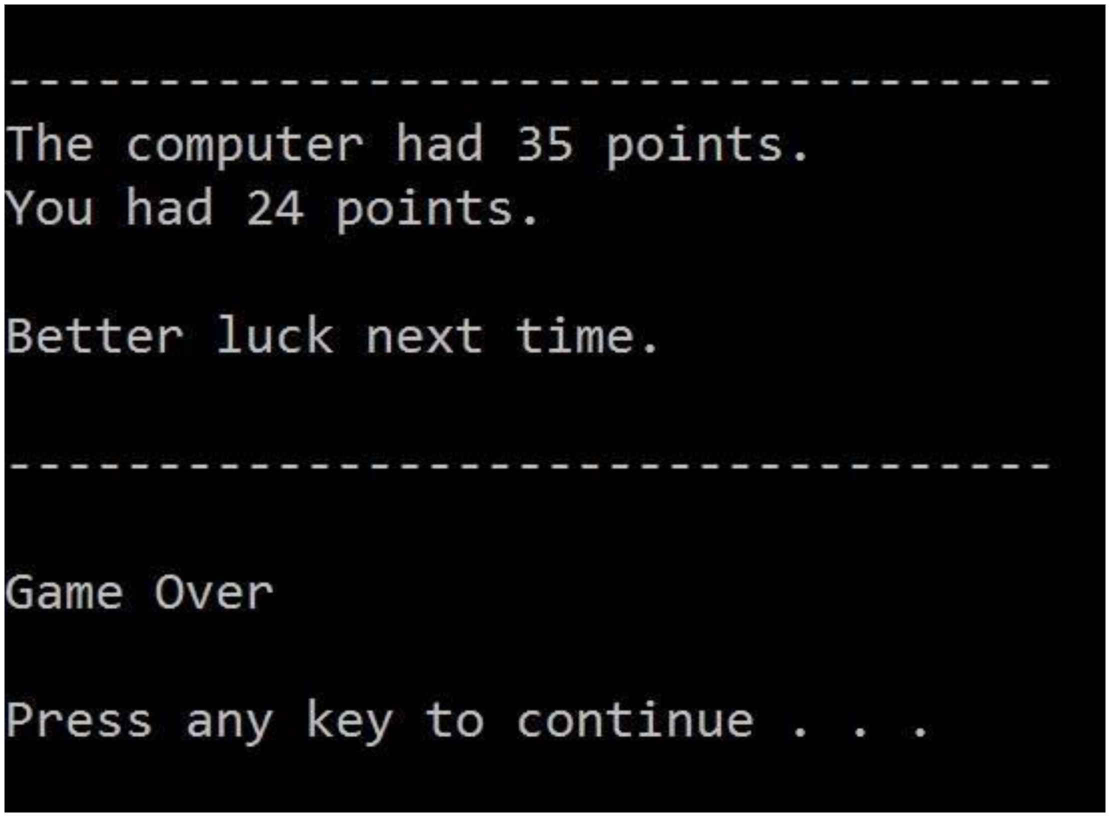


```
-----------------------------------

Game Over

Press any key to continue . . .
```

Sample run 2:

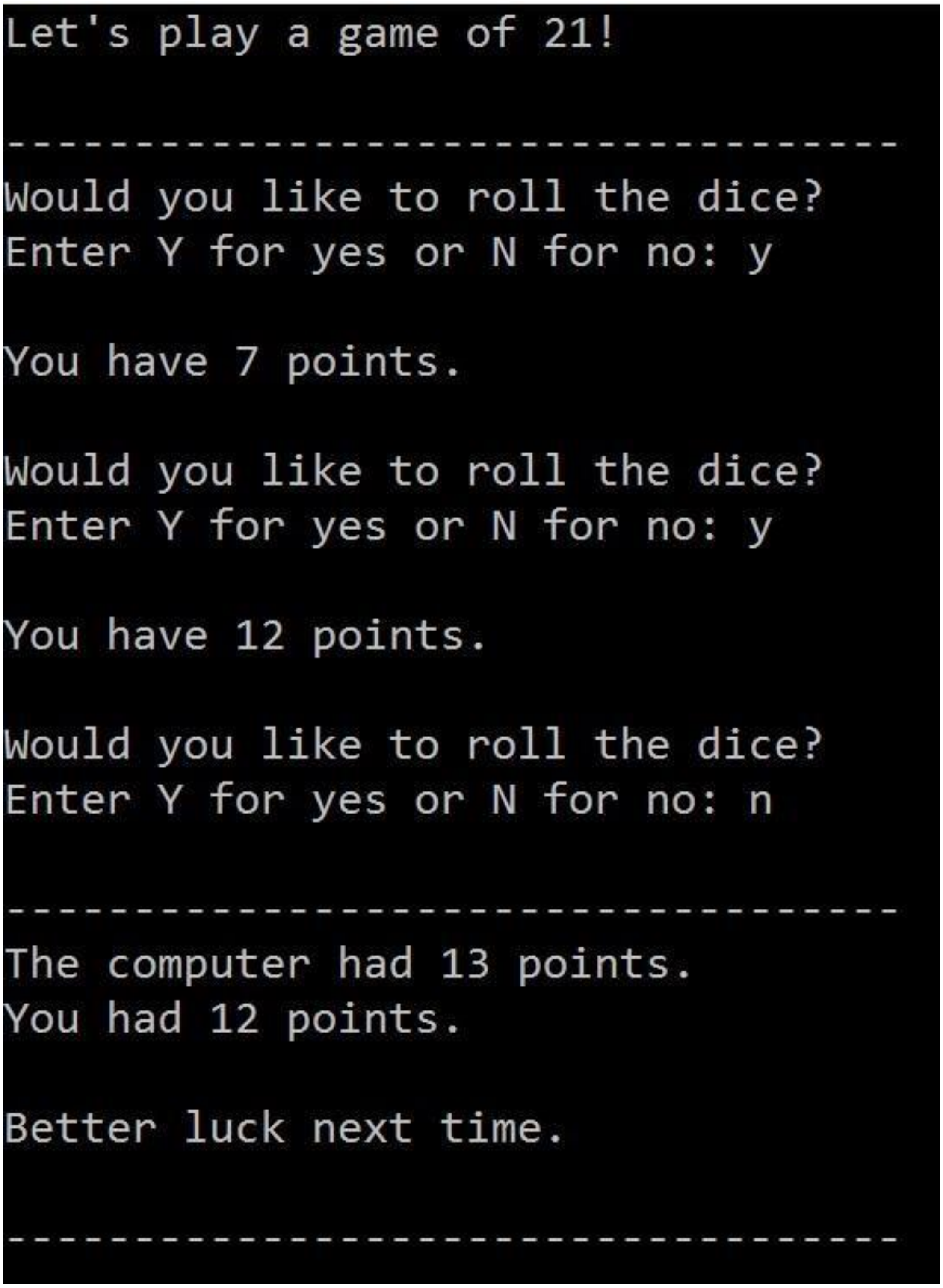


```
Let's play a game of 21!

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

Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 7 points.

Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 13 points.

Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 17 points.

Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 24 points.
```

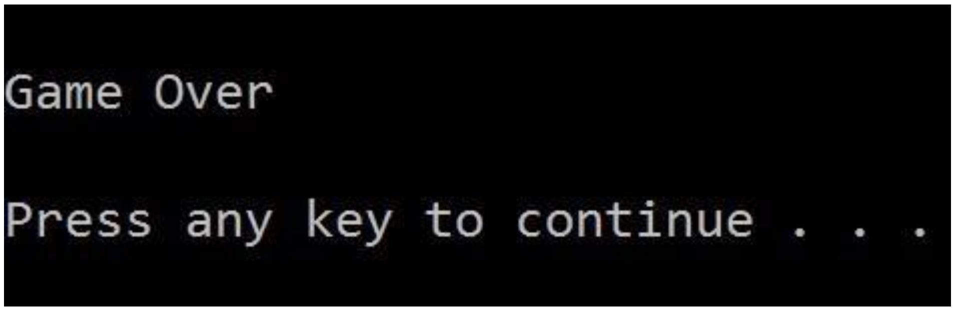


```
-----------------------------------
The computer had 35 points.
You had 24 points.

Better luck next time.

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


Game Over

Press any key to continue . . .
```

Sample Run 3:

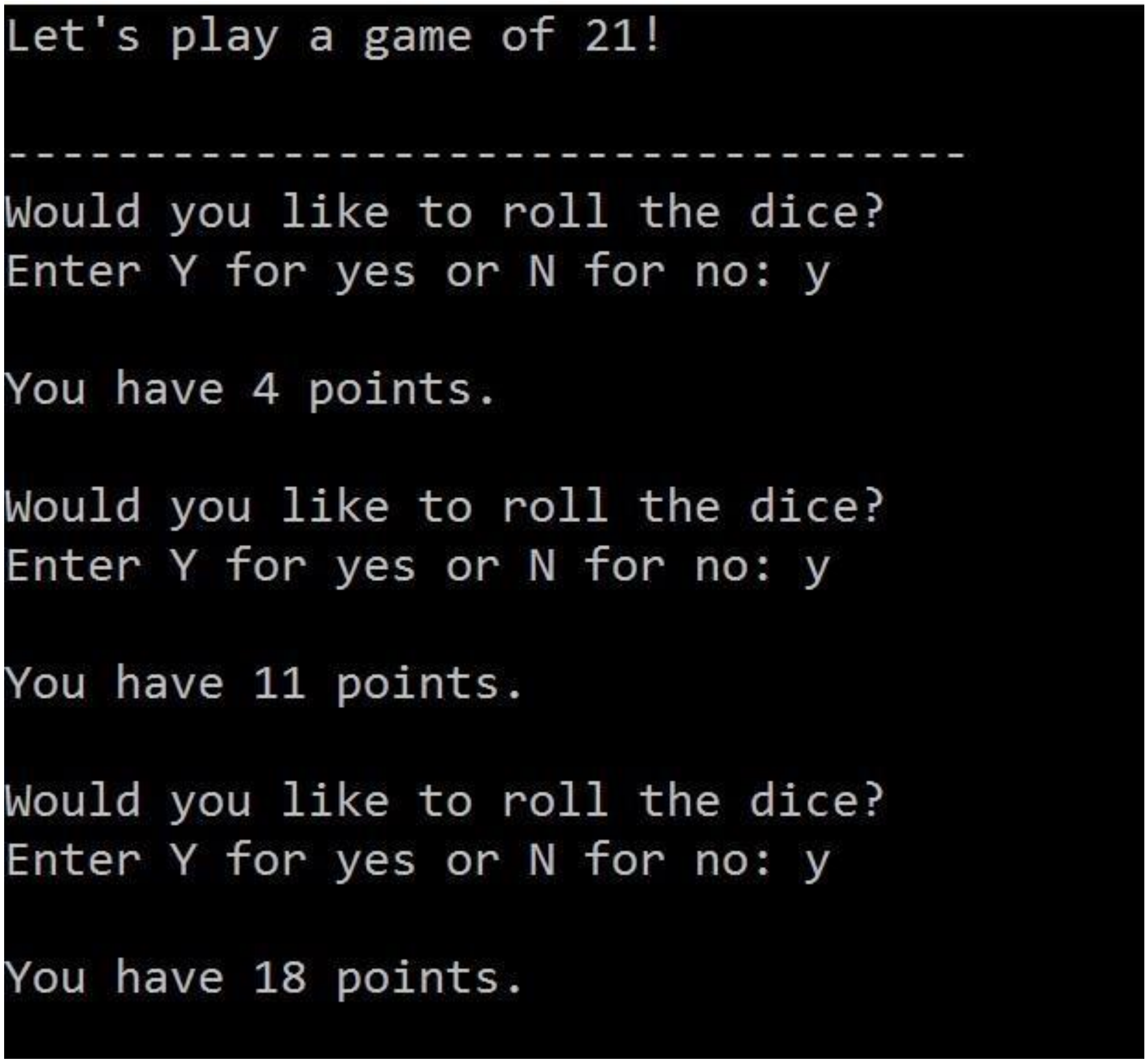


```
Let's play a game of 21!


-----------------------------------
Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 7 points.

Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 12 points.

Would you like to roll the dice?
Enter Y for yes or N for no: n


-----------------------------------
The computer had 13 points.
You had 12 points.

Better luck next time.


-----------------------------------
```

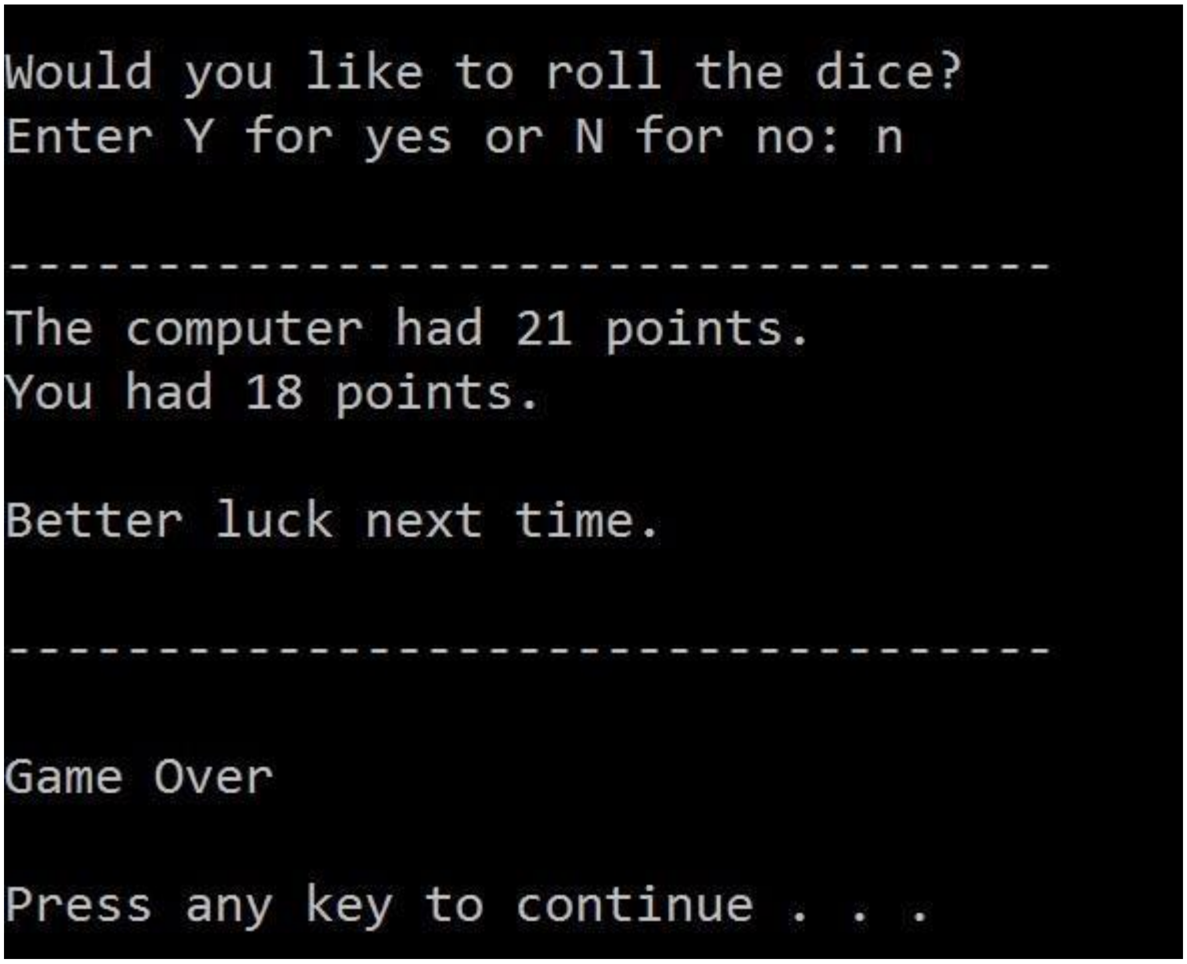


```
Game Over

Press any key to continue . . .
```

Sample Run 4:

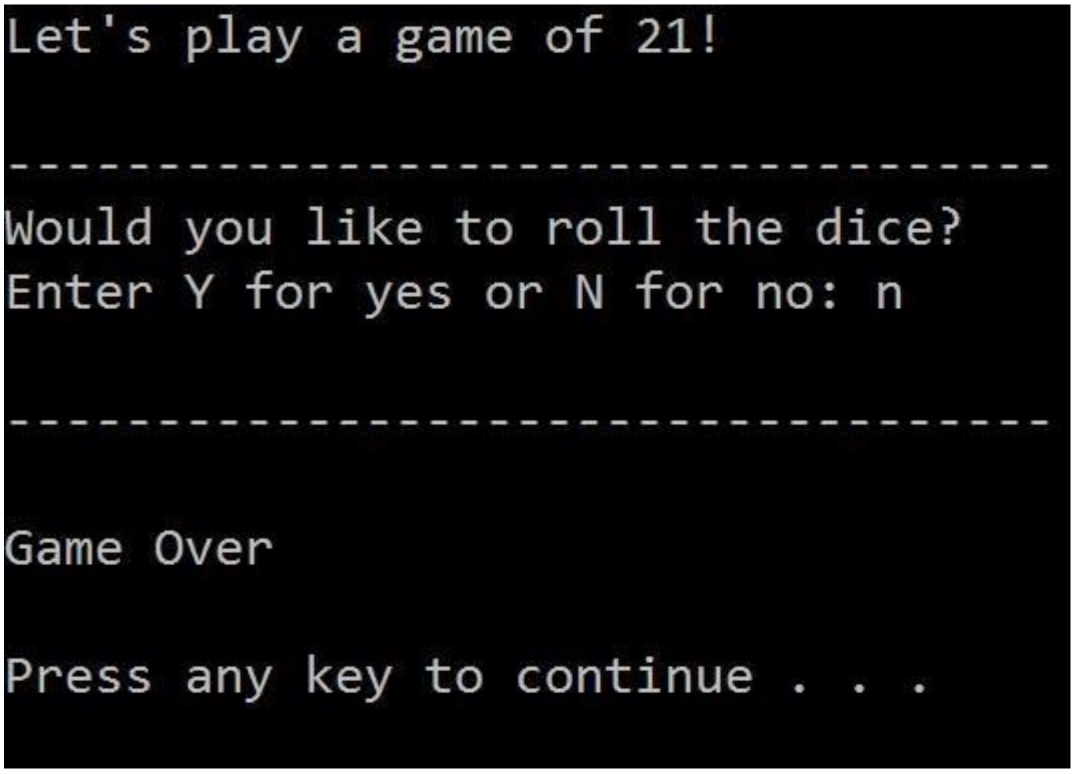


```
Let's play a game of 21!

-----------------------------------
Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 4 points.

Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 11 points.

Would you like to roll the dice?
Enter Y for yes or N for no: y

You have 18 points.
```



```
Would you like to roll the dice?
Enter Y for yes or N for no: n


-----------------------------------
The computer had 21 points.
You had 18 points.

Better luck next time.


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


Game Over

Press any key to continue . . .
```

Sample Run 5:



```
Let's play a game of 21!


-----------------------------------
Would you like to roll the dice?
Enter Y for yes or N for no: n


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


Game Over

Press any key to continue . . .
```