CSCI 1301 - Lab 18

Clément Aubert

March 15, 2018

Part I - Reading Part

Read the Part II of the Homework #5, available at http://spots.augusta.edu/caubert/teaching/2018/spring/csci1301/hw/05.pdf.

Part II - Guide For Problem 1

As you may notice, your program should essentially be one big loop. Let us try to make a program that would have a similar behaviour, shall me?

Hint 1 - Repeating

Write a program that asks the user for an integer, add that integer to all the previous integers entered so far, and repeat as long as the sum of all the integer entered so far is less than 100. A `do while` loop will be more appropriate for this task, something *like*:

```
int sum = 0;
do
{
    Console.WriteLine("Enter an integer.");
    // Here, we need some code to read from the user.
    // And some code to sum the values entered so far.
} while (sum < 100);
```

Hint 2 - Converting the User Input

Now, in the problem, we don't want to ask the user for actual integers. We want to give the user the possibility to enter a character, and then to map that character to a value. Write a small program that

1. Declare a double variable, and a string variable.
2. Ask the user to enter "N", "D" or "Q", and assign the value entered to that string variable.

3. Assign 0.05 to the double if the string entered was "N", 0.10 if the string entered was "D", and 0.25 if the string entered was "Q".

To Conclude...

The program implementing problem 1 is basically taking the two previous hints, and putting them together!

Make sure you don't overlook the other aspects of the problem: to return the change, and once this is working, to implement the additional features.

Part III - Guide For Problem 2

Hint 1 - Generating Random Numbers

The most surprising part in this problem is probably the use of random numbers. This is the way we will simulate some really primitive artificial intelligence: the computer will play rock-paper-scissors by just randomly taking one of the possibilities.

The code given in the problem (without the comments) is the following:

```
Random myRandomObject = new Random();
int a, i = 0;
while (i <= 100)
{
    a = myRandomObject.Next(1, 11);
    Console.Write(a + " ");
    i++;
}
```

1. Compile it, execute it, and observe what is displayed on the screen.
2. Are you able to modify this code, so that the values randomly generated are between 1 and 3?
3. Can you write a program that asks the user if (s)he wants to generate a new value, and display a new random number between 1 and 3 every time (s)he answers "Yes", and quit otherwise?

Hint 2 - The Logic of The Game

Complete the following table with **W**in, **L**oose or get a **T**ie:

If you play	and the computer plays	then you
Rock	**Rock**,	_____
Rock	**Paper**,	_____
Rock	**Scissors**,	_____

If you play	and the computer plays	then you
Paper	**Rock**,	_____
Paper	**Paper**,	_____
Paper	**Scissors**,	_____
Scissors	**Rock**,	_____
Scissors	**Paper**,	_____
Scissors	**Scissors**,	_____

Now, in our program, the user will enter a string made of a single character ("R", "S", "P") and the computer will randomly generate an integer (1, 2, 3), so we can't "directly" use this table in our program.

But, exactly as you associate a string made of only one character to a word (R $\leftrightarrows$ Rock, P $\leftrightarrows$ Paper, S $\leftrightarrows$ Scissors), you can associate an integer to a word, can't you? Then, how to decide who won or if it is a tie should be easy!

To Conclude...

Writing the program for this game is pretty much putting those two hints together, and enclosing the whole in a `while` loop. For this part, looking at the first hint of Problem 1 could be of some help.