COMP20220 Programming II (Conversion)

Test 2

Answer both Questions 1 and 2.

Assignment due date: Sunday, 6^{th} March, 23:50 hrs.

Submissions:

- Copy your **.java** (not **.class**) program files into a directory.
- Name this directory `Firstname_Lastname_StudentNumber_Test2`.
- Zip this directory and submit via Moodle.

Late submission penalties:

- Penalised by 3 percentage points each day late. For example, for an assignment worth 100 marks, the maximum mark is 97 if one day late, 94 if two days late, etc.
- Assignments submitted more then 2 weeks late will not be accepted/graded.

Please note:

- This is not a team/group assignment. Each student must submit her/his own work.
- Please ask if you have any questions about this. See the course Moodle for information on the UCD plagiarism policy.

Question 1

Write a *hangman* game that randomly generates a word and prompts the user to guess one letter at a time, as shown in Figure 1. Initially, each letter in the word is displayed as an asterisk. When the user makes a correct guess, the actual letter (or letters, if the correctly guessed letter occurs more than once in the word) is then displayed. When the user guesses all the letters in the word, display the number of incorrect guesses and ask the user to enter 'y' (or 'Y') to play the game again, or any other character to exit.

In each game, the word to be guessed should be randomly selected from an array as follows:

```
String[] dictionary = {"hello", "world", ...}; // Include at least two words in the array
...
int index = (int) (Math.random() * dictionary.length);
String target = dictionary[index];
```

See Figure 1 for an example of a sample run of the program. Your program should display messages to the user **exactly** as per the format shown. Some points to note:

- If the user guesses a letter which was previously guessed, and if the letter is present in the word, the program should prompt the user as shown and this guess should not count as an incorrect guess.
- If the user guesses a letter which was previously guessed, and if the letter is not present in the word, the program should count this guess as an incorrect guess.

```
Guess a letter [*****] > r
        'r' is not in the word
Guess a letter [*****] > l
Guess a letter [**ll*] > s
        's' is not in the word
Guess a letter [**ll*] > l
        'l' is already in the word
Guess a letter [**ll*] > r
        'r' is not in the word
Guess a letter [**ll*] > h
Guess a letter [h*ll*] > o
Guess a letter [h*llo] > e
         You win! The word is [hello]. You missed 3 times.

Enter 'y' to play again or any other character to exit > y
Guess a letter [*****] >
```

Figure 1: Sample run of the program. The word to be guessed is `hello`.

Hint:

- One approach is to use a `char` array to represent the word to be guessed. Initialise all elements to `'*'` and update the element(s) in the array each time the user guesses a letter correctly.

Question 2

The bean machine, also known as the Galton box, is a device for statistics experiments named after English scientist Sir Francis Galton. It consists of an upright board with evenly spaced pegs in a triangular form, as shown in Figure 2.

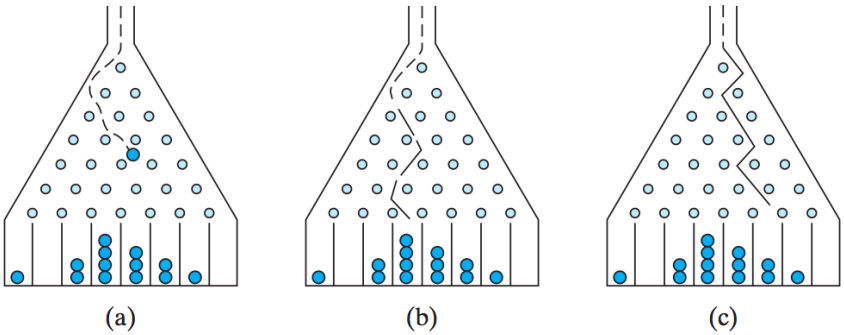

Figure 2: Each ball takes a random path and falls into a slot.

Balls are dropped from the top of the board. Every time a ball hits a peg, it has a 50% chance of falling to the left or to the right. The piles of balls accumulate in the slots at the bottom of the board.

Write a program that simulates the bean machine. Your program should prompt the user to enter the number of the balls and the number of the slots in the machine. (Note that the number of pegs that a ball hits along its path is always given by the number of slots minus 1.) Simulate the falling of each ball by printing its path. For example, the path for the ball in Figure 2(b) is LLRRLLR and the path for the ball in Figure 2(c) is RLRRLRR.

Your program should display the path taken by each ball and print the number balls in each slot **exactly** as per the format shown in the following sample run:

```
Enter the number of balls to drop: 2
Enter the number of slots (miniumum 2): 8

Paths:
RLRRLRR
RLRLRLL

Number of balls in each slot:
0 0 0 1 0 1 0 0
```

Figure 3: Sample run of the program.

Use the following method in your program:

```
// Generates one random path.  Returns the path and updates the count (in array slots)
// of the slot the ball falls into.
public static String generateRandomPath(int numberOfSlots, int[] slots)
```

Hint:

- Create an array named `slots`. Each element in `slots` stores the number of balls in a slot. Each ball falls into a slot via a path. The number of Rs in a path is the position of the slot where the ball falls. For example, for the path RLRRLRR, the ball falls into `slots[5]`, and for the path is RLRLRLL, the ball falls into `slots[3]`.