ACS-1903
Assignment 2
Due by Friday November 5 at 11:59 pm

- Submit your .java files via Nexus
- Include your name and student number in each file as a comment

Note: concepts that have not been covered in class cannot be used, for example arrays, BufferedReader, String methods that have not been introduced, the use of break other than in switch statements, regular expressions, System.exit(), a return in the main method, etc. Do not code loops with forms like `while(true)`.

Check with your instructor or TA if you have concerns.

Use meaningful names for methods and variables; you must follow the CamelCase naming standard.

Ensure you have followed standard indentation practices – consider using the Auto-layout feature in BlueJ. Avoid excessive use of blank lines.

1. (15 marks)

   Write a program **DNAStrand** so it obtains a DNA strand and then displays:

   a. the number of times each base appears
      For instance, if the user inputs AAGC then the output is
      2 Adenine
      0 Thymine
      1 Cytosine
      1 Guanine

   b. the complement of the DNA strand
      Chargaff's rule, also known as the complementary base pairing rule, states that DNA base pairs are always adenine with thymine (A-T) and cytosine with guanine (C-G). That is,
      - each A is replaced with a T and each T is replaced with an A.
      - each C is replaced by a G and each G is replaced by a C.

      Knowing this rule, you can determine the complementary strand to a single DNA strand based only on the sequence.

   Include a method named `displayResults(…)` that has a parameter list of all the values needed to display the results.

Sample output:

```
Enter a DNA strand:
ATGATTGACATTGAGGATCCAT
7 Adenine
7 Thymine
3 Cytosine
5 Guanine
Complementary strand: TACTAACTGTAACTCCTAGGTA
```

2. (15 marks)

Consider this scenario that arises in a certain board game.

One player (the attacker) is going to battle another player (the defender). Each player will roll a six-sided dice. These are fair dice in the sense that each of the six sides is equally likely to appear. However, the two dice are different:

The attacker's dice is red, and has the numbers 1, 2, … 6 on the sides. That is, this is a standard six-sided dice.

The defender's dice is blue, and has six sides with the numbers 1, 2, … 5 on the sides. What is noteworthy is that the number 5 appears on two sides.

Each player rolls his/her dice and the player who rolls the larger number is the winner. If they tie (both roll the same number) they roll again, repeating until there is a winner.

Write a program named **AttackerVsDefender** to give us a sense of what advantage the attacker has in this scenario.

Simulate the scenario 100,000 times. Display information (see below) regarding the simulation:

Sample output:

```
#Wins for attacker: 53283
#Wins for defender: 46717
```

Based on this simulation, the attacker is 1.1405484085022584 times more likely to win a battle.

3. (10 marks)

   Consider a flight number to have the following valid formats:
   `AA999` `AA999-999`
   where A represents a letter and 9 represents a digit.

   For example:
   Valid flight numbers: WS697, AC889-129
   Invalid flight numbers: W1234, WSJ12, 22222, AC889129, AC889-1239
   Write a program named **FlightValidation** that validates a flight number entered by the user where the user might have entered either of the valid formats above.

   Include the following methods:
   - `validateFlightNumber(…)` which accepts the flight number and returns a boolean that indicates whether or not the entry is a valid flight number.
   - `displayResult(…)` which accepts the flight number and boolean, and displays the result.

4. (10 marks)

   Write a program named **AnalyzeCode** that reads a java file and displays the total number of imports, control structures, and statements.

   Sample output:
   ```
   imports: 3
   control structures: 8
   statements: 17
   ```

   Notes:
   - Control structures include if, while, for, do-while, switch
     - Consider that some control structures may or may not be written with a space after the keyword i.e. `while()` vs `while ()`
   - A statement ends with a semicolon ;
   - Submit your code that reads the program itself (AnalyzeCode) but test by reading in other programs.

Submit your files (`DNAStrand.java, AttackerVsDefender.java, FlightValidation.java, AnalyzeCode.java`) via Nexus.