EECS 1022 Labtest 2 Version A

Friday November 16, 2018, 10:00-11:30

Instructions

This labtest is closed book, no aids allowed. You must work individually. You are not allowed to browse any documents on the web other than those provided in this labtest page.

There is a set of multiple-choice questions and a set of programming questions. You will fill your answers to all these questions into a Java class called `Utilities.java` which you will then submit. Here is how to proceed:

- Launch Android Studio and open the **existing project** called `Labtest2A`.
- Select the Java class `Utilities.java` in the editor window; it contains the starter code.
- Then go over the questions below and enter your answers into the `Utilities.java` file and save your changes.
- Test your solutions by running the `UtilitiesTester.main()` method (right click on the `UtilitiesTester.java` class in the project window and select to run `UtilitiesTester.main()`). Your output should be as shown here.
- When you are ready, use Web Submit to submit your modified `Utilities.java`. To do this, first open the link https://webapp.eecs.yorku.ca/submit in your browser, login using your Passport York account, select your course `1022`, select the assignment `labtest2a`, navigate to and choose the file `Utilities.java` (it should be in folder `AndroidStudioProjects/Labtest2A/app/src/`), and then click the "Submit files" button at the bottom of the page. If you have successfully completed this task, you will see a message saying that you have submitted the file `Utilities.java` with the submission time.
- You can submit as many times as wou want and only the last version submitted will be graded.
- **You must submit your answers before the labtest period ends at 11:30; submission will be disabled after that time.**
- **Check to make sure that your submitted file appears in the list of submitted files in WebSubmit. Also check that you have submitted the right version (you can download it and open it to check).**
- **Make sure that your `Utilities.java` class compiles without errors. If there are compilation errors, you may get a grade of 0.**
- Here, you can find a table of common Java regular expressions constructs.
- The Java API is here .

If you finish early, you may leave the lab. After the labtest ends, the lab will remain open until the end of the lab period and you may work on Lab 5, which is described in Chapter 5 Collections - Doing of the textbook.

Multiple Choice Questions

For these questions, enter your answers into the `Utilities.java` file by editing the method corresponding to the question, replacing the `?` in the printed string by your answer, a, b, c, d, or e.

Question 1

Which Java loop construct ensures that the *body* is always executed at least once?

a. `for (initial; condition; bottom) { body }`,
b. `while (condition) { body }`,
c. `do { body } while (condition)`,
d. all of the above,
e. none of the above.

Question 2

Consider the following Java code:

```
int r = 0;
if (x > 4) {
   if (y <= 10) {
      r = 1;
   } else {
      r = 2;
   }
}
else
{
   if (y >= 15) {
      r = 3;
   } else {
      r = 4;
   }
}
```

Assume that `x` has the value `3` and `y` has the value `15`. What is the value of `r` after the code has been executed?

a. `0`,
b. `1`,
c. `2`,
d. `3`,
e. `4`.

Question 3

Consider the following Java code:

```
for (int i = 11; i > 5; i = i - 2) {
   r = r + i;
}
```

Assume that `r` has the value `0` initially. What is the value of `r` after the code has been executed?

a. `0`,
b. `32`,
c. `27`,
d. `20`,
e. none of the above.

Question 4

Which of the following sets of strings does the regular expression `^a[ab]c?$` match?

a. `{ "aa", "ab", "aac", "abc"}`
b. `{"aac", "abc"}`
c. `{ "a", "aa", "ab", "aac", "abc"}`
d. `{ "aab", "aabc" }`
e. none of the above.

Question 5

Which of the following strings does *not* match the regular expression `^[ab]*$`

a. `""`
b. `"ba"`
c. `"aba"`
d. `"bab"`
e. `"aca"`

Programming Questions

Question 6

Go to the `Utilities.java` class and implement the method

```
double ticketPrice(int age)
```

Given the age of the customer `age`, return the correct movie ticket price for the customer.

The ticket price should be 8.50 if the age is less than or equal to 12 or greater than or equal to 65, and 12.50 otherwise.

You can assume that all input values of `age` are greater than or equal to 0. There is no need to check for it.

Your returned double value may be within ± 0.1 of the expected output.

You can see some examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 7

In the `Utilities.java` class, implement the method

```
double sumReciprocals(int n)
```

Given a positive integer `n`, return the sum of the reciprocals of all positive integers less than or equal to `n`.

For example, when `n` has the value 3, the returned value should be 1.8333 = 1/1 + 1/2 + 1/3.

It is not required to validate the input `n`.

Your returned double value may be within ± 0.1 of the expected output.

You can see some examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 8

In the `Utilities.java` class, implement the method

```
String makeRepetitions(int n, String s)
```

Given a natural number `n` and a string `s`, it returns a string that contains `s` repeated `n` times.

For example, when `n` has the value 3 and `s` has the value `"abc"`, the returned value should be `"abcabcabc"`.

You can assume that the input argument `n` is a natural number; it is not necessary to validate it.

You can see some examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 9

In the `Utilities.java` class, implement the method

```
String removePrefix(String s1, String s2)
```

Given a string `s1` and a string `s2`, find the first occurrence of `s2` as a substring in `s1` and return `s1` with all characters before the first occurrence of `s2` removed; if `s2` does not occur as a substring in `s1`, return `s1` unchanged.

For example, when `s1` has the value `"abcdeabcde"` and `s2` has the value `"cd"`, the returned value should be `"cdeabcde"`.

You can see more examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 10

In the `Utilities.java` class, implement the method

```
String allPos(String s1, String s2)
```

Given a string `s1` and a string `s2`, return a string containing the positions of all occurrences `s2` in `s1` separated by a comma. If there are no occurrences `s2` in `s1`, return the empty string.

For example, when `s1` has the value `"abcdeabcde"` and `s2` has the value `"cd"`, the returned value should be `"2,7"`.

You can see more examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.