UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

Assignment 1

Algorithms, Data, Expressions, Variables, and I/O

Date Due: September 20, 2019, 6:00pm **Total Marks: 26**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form **aNqM**, meaning Assignment N, Question M. Put your name and student number at the top of every document you hand in. These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you. Do not submit folders, zip documents, even if you think it will help.
- Programs must be written in Python, and the file format must be text-only, with the file extension `.py`.
- Documents submitted for discussion questions should make use of common file formats, such as plain text (`.txt`), Rich Text (`.rtf`), and PDF (`.pdf`). We permit only these formats to ensure that our markers can open your files conveniently.
- **Assignments must be submitted electronically to Moodle.** There is a link on the course webpage that shows you how to do this. Your TAs will cover this in the first tutorial.
- **Moodle will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Questions are annotated using descriptors like "easy" "moderate" and "tricky". All students should be able to obtain perfect grades on "easy" problems. Most students should obtain perfect grades on "moderate" problems. The problems marked "tricky" may require significantly more time, and only the top students should expect to get perfect grades on these. We use these annotations to help students be aware of the differences, and also to help students allocate their time wisely. Partial credit will be given as appropriate, so hand in anything you've done, even if it's not perfect.

Question 1 (6 points):

Purpose: To familiarize you with the Academic Honesty policy, and Moodle's assignment submission, and to do a little writing practice.

Degree of Difficulty: Easy

Read the Academic Honesty webpage at `https://www.cs.usask.ca/students/current-students/academic-honesty.php` (click coloured text to open). In the context of what you read, which of the following scenarios are violations of academic honesty? Explain in one or two sentences why or why not by referring to the academic honesty webpage. Please do not write lengthy answers.

(a) Red and Blue are working on an assignment together. Red emails her answer to Blue. Blue copies it and hands it in.

(b) Giovanni and Ash are enemies. Ash hacks into Giovanni's user account, and copies Giovanni's work for the assignment.

(c) Squirtle, Charry, Bulba, and Pikachu are a study group who have decided to work together on the assignment. Each of them answered one question, and then brought the answer to the group for discussion. After a long process of discussion and analysis, in which all the group members participated equally, they created a final answer to each question that they all copied.

(d) Brock and Misty just met in the lab. Misty has done some programming before, but Brock is a novice. Misty helped Brock fix his program so that it looked pretty much like her own.

(e) Jessie and James were passing through the lab when their friend Meowth stopped them and asked them for help figuring out why his program wasn't working right. James and Jessie looked at Meowth's program and explained what the problem was and what was causing it. Meowth pretended to thank them, and proceeded to fix the error on his own.

(f) Mewtwo and Lucario worked on the assignment together. Once they were fairly sure they understood the solutions, they destroyed all copies of their work. After playing an hour of Pokemon, they recreated the solutions to the assignment separately, from scratch, without consulting each other, or the notes they made.

Feel free to discuss these scenarios with other students, TAs, and instructors. It's very important that everyone understand the rules, as they will be applied to all assignments in CMPT 141. When the solutions are released, come back to these scenarios to make sure your understanding is consistent with ours!

What to Hand In

Hand in your answers in a file called `a1q1`. Allowed file formats are plain text (`.txt`), Rich Text (`.rtf`), and PDF (`.pdf`). We permit only these formats to ensure that our markers can open your files. Documents that cannot be opened conveniently will not be graded and receive 0 points.

Evaluation

- 1 mark for each correct answer. Answers should be justified by referring to the academic honesty policy. Writing style need not be formal, but should reflect an attempt at university-level English.

We are not grading for grammar or spelling. These are important, but there are other classes to evaluate that. However, there should be an attempt to make complete sentences, and students should avoid unprofessional abbreviations. Answers that cannot be understood because of poor writing will receive 0 marks.

Question 2 (3 points):

Purpose: To practice the concept of *stepwise refinement*.

Degree of Difficulty: Easy

Many students are too poor to have a dishwasher, so they might need to wash dishes by hand. Here is a pseudocode algorithm for the task of manually washing dishes.

Rewrite the algorithm in more detail by using stepwise refinement. Replace each action with at least three sub-actions that describe how to accomplish the original action. There are many possible correct answers. Any reasonable, sensible answer will be accepted.

```
Algorithm WashDishes:

fill sink
wash dishes
dry dishes
```

What to Hand In

Hand in your answers in a file called `a1q2`. Allowed file formats are plain text (`.txt`), Rich Text (`.rtf`), and PDF (`.pdf`).

Evaluation

- 1 mark for each original action that was refined acceptably.

Question 3 (3 points):

Purpose: To understand the parts of an algorithm.

Degree of Difficulty: Moderate

```
1  Algorithm checkGrades:
2  Input: a set of numbers representing percentile student grades.
3       Grades range from 0 to 100, where 50 is a passing grade.
4  Output: a variable 'fraction' that is the percentage of grades
5       that are at least 50 or higher
6
7  let count = 0
8  for each grade in the set of grades:
9    if the grade is at least 50:
10      let count = count + 1
11
12 let fraction = count / the size of the set of grades
```

Answer the following questions about the above algorithm:

(a) How many **inputs** does this algorithm have? What are these input(s)?

(b) How many **outputs** does this algorithm have? What are these output(s)?

(c) In as **few words as possible**, what is the **problem** that the algorithm is solving?

What to Hand In

Hand in your answers in a file called `a1q3`. Allowed file formats are plain text (`.txt`), Rich Text (`.rtf`), and PDF (`.pdf`).

Evaluation

- 1 mark for each question

Question 4 (5 points):

Purpose: To practice console input and output of strings.

Degree of Difficulty: Easy

A "mad-lib" is a fill-in-the blank game. One player writes a short story in which some words are replaced by blanks. For each word that is removed, the appropriate part of speech is noted: e.g. noun (person/place/thing), adjective (word that describes a noun), verb (an action, e.g. eat), adverb (modifies a verb, e.g. quickly). Then, before reading the story, the story-writer asks the other player to write down a word of the appropriate part of speech for each blank without knowing the context in which it will be used. In this way, a humorous (sometimes) or non-sensical (usually) story is created.

Write your own mad-lib. It must have **at least three blanks** in it. Now write a Python program that does the following:

(a) Using the `input()` syntax (see textbook Section 2.4.2 "Reading Strings from the Keyboard") for reading strings from the console, prompt the user to enter a word of the appropriate part of speech for each blank in your program. Have a different, appropriately named variable refer to each word.

(b) Print your story to the console using the `print()` syntax, filling in the blanks in your story using the strings referred to by the variables that you gathered in Step 1. While it is possible to do this using a single `print()`, readability of code is important! You are permitted to use as many `print()` statements as you wish.

You must write your own unique story. There is no good reason why two students should submit the same story. Remember you must use a minimum of three blanks, and therefore, your program must read at least three words from the console. You can have more blanks, but get all your other work done before you spend a lot of time having fun with this question!

Sample Run

Here is an example of how your program's console output might look. **Do not submit this story!** Write your own story. In our example, we use green text to show what the user typed in; blue text highlights where the user's data gets put in the story. If you're using PyCharm to run your program, the user input will also be green, but it won't show any blue text.

```
Enter a verb ending in "ing":  zipping
Enter a noun:  squirtle
Enter a verb (past tense):  zapped
Pikachu was zipping through some tall grass.
Suddenly, a squirtle appeared!
The squirtle zapped Pikachu.
It was not very effective.
```

What to Hand In

Hand in your solution in a file called `a1q4.py`.

Evaluation

- 2 marks for reading text from the console;
- 2 marks for printing the story to the console in an aesthetically pleasing manner;
- 1 mark for choosing appropriate variable names (variable names should be descriptive of the data referred to by the variable; single-letters and abbreviations should be avoided).

Question 5 (3 points):

Purpose: To practice console I/O, particularly reading numbers from the console, and use of variables.

Degree of Difficulty: Easy

We've all fantasized about destroying our computers when they don't work properly. In the spirit of scientific inquiry, let's write a program to compute how long a computer will take to hit the ground when dropped out of a window. Ignoring air resistance, the time (in seconds) when the computer hits the ground is:

$$t = \sqrt{(2 \times H/9.8)}$$

In the above formula, t is the time in seconds, and H is the height in meters from which the computer is dropped with no initial velocity. The formula may not be accurate if you are throwing the computer!

Write a Python program that does the following:

(a) Put the following as the first line of your program:
`import math as m`
We'll cover what this means in class a bit later. For now, it is enough to know that this line allows us to use mathematical functions defined in the module called math.

(b) Prompt the user to enter the starting height of the computer in meters. Read the user's response **as a number** using the `input()` syntax (see textbook Section 2.4.3 "Reading Numbers from the Keyboard") and set a variable called `H` to refer to it.

(c) Compute the time at which the computer hits the ground using the above formula and have a variable called `time_of_impact` refer to the result. You can compute the square root of the value of an expression by writing `m.sqrt(`*expression*`)`. For example, the square root of two times seven can be written `m.sqrt(2*7)`. This syntax is made available by the line you wrote in step (a): we're using the square-root function defined in the math module.

(d) Use the values of the two variables to display a message to the console that describes the solution to the problem (see sample program run, below).

Sample Run

If you've completed your program correctly, you should see something like the following on the console when you run it:

```
From what height is the computer dropped?  42
There is a satisfying explosion of circuitry!
The computer dropped from 42.0 meters hits the ground after 2.9277002188455996 seconds.
```

Note: the green number is text entered by the user at the console; blue text highlights the values referred to by program variables `H` and `time_to_impact`. Of course, you'll see different numbers if you enter different values of `H`. If you are using PyCharm, the input text will be green (unless you change the preferences), but none of Python's console output will be blue.

What to Hand In

Hand in your solution in a file called `a1q5.py`.

Evaluation

- 1 mark each for parts (b) through (d). The solution can be as short as three or four lines of code (not counting part (a))!

Question 6 (6 points):

Purpose: To practice the use of arithmetic expressions.

Degree of Difficulty: Moderate

Phoenix Wright is a lawyer who runs a small law office with some number of assistants. When (if?) Phoenix gets paid for a case, he divides the money between himself and all his staff. The office contract states that Phoenix, as senior partner, takes 25% of the fee for himself and divides the rest equally among all his staff.

Write a program for Phoenix that will ask the user to input the payment for one case (in dollars), and the number of staff in the office (not including Phoenix himself). Then the program should display the value of Phoenix's share of the fee, the value of the remaining share to be divided amongst the staff, and the value of the pay that each staff member takes home.

You may assume that the user supplies valid input from the console, that is, a positive number for the fee, and a positive number for the staff size.

Sample Run

If completed correctly, your program's console output should look something like this. As usual, green text shows the text entered by the user, and blue text highlights the values calculated by the program.

```
Enter the payment for the case:  9000
Enter the number of staff:  2
Phoenix Wright's 25% share of the fee is worth: $2250.0
The staff's 75% share of the fee is worth: $6750.0
Each staff member takes home: $3375.0
```

Notice that the user is not expected to type the $ when entering the value of the legal fee. It is also not a concern if your displayed dollar amounts have more or less than two digits after the decimal place, or don't have a decimal at all. You will not be penalized for this (unless the values computed are incorrect). If you are having trouble getting the dollar amounts to print right next to the dollar signs without a space in between, think about how you might use the string concatenation operator (the very end of textbook Section 2.3.4 "Operators on Strings") to achieve the desired output.

What to Hand In

Hand in your solution in a file called `a1q6.py`.

Evaluation

- 1 mark for reading in the total fee and number of staff from the console;
- 3 marks for displaying the correct dollar amounts (1 mark each);
- 1 mark for formatting the dollar amounts without a space between the dollar sign and the amount;
- 1 mark for choosing appropriate variable names (variable names should be descriptive of the data referred to by the variable; single-letters and abbreviations should be avoided).