COMP 1631 Assignment 1 (“Problem Solving and Algorithms”)

Given: Monday, September 14, 2020
Electronic Copy Due: Monday, September 21, 2020 before 11:59 p.m. sharp
NO LATES ACCEPTED!

Objectives

- To build problem solving skills.
- To gain experience following Polya’s four steps.
- To learn about algorithms by analyzing or developing a few.
- To practice using Linux and the Emacs editor.
- To learn how to submit an assignment.

Grading Information

Your work will be graded according to:

- The quality of your explanations: they must be expressed clearly and logically with correct spelling and grammar;
- The quality of your algorithms (questions 3 and 4 only): they must be clear and precise sequences of instructions **and** must use suitable notation (e.g. pseudocode)

General Instructions

This assignment deals with problem solving and the development of algorithms. Although none of the problems involve C++ programming, the skills you begin developing now will help you design and code programs later in the course.

Solving a problem systematically entails following Polya’s four steps, which are:

1. Understand the problem (Analysis)
2. Devise a plan (Design)
3. Carry out the plan (Implementation)
4. Look back (Review)

Using `Linux` and `Emacs`

Your assignment must be prepared using the `Emacs` text editor on `INS`. Use `Emacs` to create a file called “`asg1.txt`”. Handwritten work or solutions prepared using alternative tools will not be accepted. Please follow the detailed submission instructions below.

Final Instructions

Get in the habit of starting assignments early and then working on them a little bit each day. If you establish this good habit now, later you’ll be glad you did.

This assignment is to be completed <u>individually</u>. Do not share your solutions with other students in the class. Doing so will rob them of the chance to develop their own problem solving skills. Please familiarize yourself with the guidelines in the "Academic Conduct" handout before you start.
If any aspect of this assignment is unclear, please do not hesitate to ask your instructor or one of the Instructional Assistants.

Submission Instructions

Once you are ready to submit "`asg1.txt`", please follow these instructions exactly:

1. `cd` into the directory which contains `asg1.txt` (if you are not there already).
2. Run the `script` program to begin a scripting session.
3. Run the pwd command.
4. Run the `ls -al` command to display the contents of the directory (that is "ell ess space dash ay **ell**", not "ell ess dash ay one").
5. Run `cat asg1.txt`.
6. Type `exit` to terminate your scripting session. A file named `typescript` will have been created.
7. Run the `submit` program, and submit your typescript file to your instructor (be sure you know your instructor's login name and your course section number before you begin).

Note that assignments submitted any other way will not be graded. Do not e-mail your work to your instructor.

It is a good idea to practice submitting well before the due date. If you encounter problems or are unsure what do to, please consult with your instructor on an Instructional Assistant *well before* the due date. Note that you can run `submit` multiple times. For each assignment a new submission will overwrite any previous submission, up until each deadline.

Problem 1 [5 marks]

A philosophy professor assigned the following problem to his class:

> One morning at sunrise a Buddhist monk began to climb a narrow path to a temple at the top of a tall mountain. He did not walk at a constant speed but stopped occasionally to eat or rest and reached the temple shortly before sunset. The next morning he descended the same path, starting at sunrise and again walking at varying rates but generally at a faster pace than his average climbing speed the preceding day. Prove that there is a spot along the path that the monk will occupy on each trip at precisely the same time of day.

Students were to solve the problem and to describe the steps they took when solving the problem.

The brightest student in the class submitted the following proof:

> Imagine two monks walking on the same day, one up and one down, both following precisely the path that the real monk had taken and both proceeding at the same rate of speed as the real monk. The two imaginary monks must meet somewhere along the path (though we can't say precisely where), and that is the spot the monk had occupied on both trips at exactly the same time of day.

As part of step 1 the student made the following **five** observations:

1) two monks walking on one day would be the same as one monk walking on two days
2) the walking speed doesn't matter
3) it would be possible to graph the time of day against location on path for the monk on the two different days
4) while the monk may walk at varying speeds and take breaks if both the total time taken to walk the path and the distance of the path are known the average walking speed can be calculated and used
5) assuming that the monk walked at 3 km/hr the first day and at 4 km/hr the second day and the path was 8 km long an exact meeting position can be calculated

For each of the FIVE observations, from above:

- Identify the specific technique from Polya's step 1 that is being used (These are the techniques specified in the 03.1 – POLYA Handout.pdf.)

Problem 2 [10 marks]

An algorithm in pseudocode is given below. Describe concisely and unambiguously:

i. the number of input value(s) and type of each value required by the algorithm,
ii. the number of output value(s) and type of each value generated (not what the output means),
iii. what the algorithm does, (i.e. a high-level description – restating each line in English is NOT a description of what the algorithm does!).

```
set count1 = 0
set count2 = 0
read number

while read was successful do
      if number mod 2 equals 0
      then
            set count1 = count1 + 1
      otherwise
            if number div 7 equals 0
            then
                  set count2 = count2 + 1
      read number
write count1
write count2
```

Problem 3 [20 marks]

It is possible to categorize triangles in three ways 1) by angles, 2) by side lengths, or 3) by both. The angle categories are right, acute or obtuse. The side categories are equilateral, isosceles or scalene.

This question deals with categorizing triangles by the largest angle and length of all three sides.

Step 1: List the valid triangle categories that are combinations of an angle and a side categories. This will require some research. *You must provide at least one reference, book or URL, that supports your categories and restrictions*. Your source(s) **must** complete validate the information you specify.

Step 2: Write an algorithm, using pseudocode, that will read in the largest angle of the triangle and the length of the three sides in the order of left, right and bottom. The input is guaranteed to be valid, i.e. the angle will be between 0 and 180 degrees, exclusively and all sides will have a length be greater than zero. Based on this input the algorithm will determine the specific type of angle-side triangle and output the appropriate term (for example: acute equilateral). The angle-side output must be written as one item – not as two distinct items, i.e. as write “acute equilateral” and not as write “acute” and then later write “equilateral”.

Problem 4 [15 marks]

Develop an algorithm, using pseudocode, which computes the Grade Point Average (GPA) for one student at MRU. The complete MRU Grading System and Grade Point Calculation can be found at https://catalog.mtroyal.ca/content.php?catoid=21&navoid=1405.

However, this question will work with simplified rules, the only grades allowed are A+ through F, i.e. no special cases such as W’s or I’s. A student’s GPA is determined by dividing the total grade points earned by the total number of credit hours attempted. Student grades are recorded as letter grades, from A+ to F, each of which corresponds to a grade point, between 4.0 and 0.0. Each course has a credit hour, which is one of 1.5, 2, 3 or 4. The total grade points earned is the sum of the course grade points times the course credit hours.

An example of this is:

```
1) Course     2) Grade    3) Course       4)          5) Total
                            Grade        Credit        Grade
                            Points       Hours         Points
CHEM 1201       B             3       x    3      =      9
ENGL 1101       C             2       x    4      =      8
ACOM 2003       D             1       x    3      =      3
INDS 2213       F             0       x    2      =      0
PHED 1239       A             4       x   _1.5_   =      6
Total Credits                         =   13.5
                                                   ___________
Total Grade Points                                =     26
Grade Point Average =    26 / 13.5 = 1.93
```

The data will consist of the number of courses the student has taken and then numeric pairs consisting of the course grade point and credit hours, for each course. From the above example pairs will be values from columns 3) and 4) from the same line. Thus, the data for the above example would be:

```
5
3     3
2     4
1     3
0     2
4     1.5
```