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 2

Design Goals, and Development Processes

Date Due: 5 February 2021, 11:59pm **Total Marks: 40**

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.
- **Assignments are being checked for plagiarism.** We are using state-of-the-art software to compare every pair of student submissions.
- 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.
- Make sure your name and student number appear 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, or zip files, even if you think it will help. It might help you, but it adds an extra step for the markers.
- Programs must be written in Python 3.
- **Assignments must be submitted to Moodle.** There is a link on the course webpage that shows you how to do this.
- **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.
- Read the purpose of each question. Read the Evaluation section of each question.

Question 1 (5 points):

Purpose: To force the use of Version Control in Assignment 2, Question 2

Degree of Difficulty: Easy

Version Control is a tool that we want you to become comfortable using in the future, so we'll require you to use it in simple ways first, even if those uses don't seem very useful. Do the following steps.

1. Create a new PyCharm project for Assignment 2.
2. Use `Enable Version Control Integration...` to **initialize** Git for your project.
3. Download the Python and text files provided for you with the Assignment, and **add** them to your project.
4. Before you do any coding or start any other questions, make an initial **commit**.
5. As you work on Question 2, use Version Control frequently at various times when you have implemented an initial design, fixed a bug, completed the question, or want to try something different. Make your most professional attempt to use the software appropriately.
6. When you are finished your assignment, open PyCharm's Terminal in your Assignment 2 project folder, and enter the command:

```
git --no-pager log
```

7. Copy/Paste the output of the above command into a text file named `a2-git.txt`.

Notes:

- Several Version Control videos are available on Moodle via the Lecture Videos link.
- No excuses. If you system does not have Git, or if you can't print the log as required, you will not get these marks. Installation of Git on Windows 10 machines is outlined in the "*Step-by-Step Windows 10 - PyCharm UNIX command-line*" video. Git is included in base installations of Linux & MacOS.
- If you are not using PyCharm as your IDE, you are still required to use Git. See the Version Control videos on Moodle that cover using Git on the command line & via GUI.

What Makes a Good Commit Message?

A good commit message should have a **subject line** that describes *what* changed, a **body** that explains *why* the change was made, and any other *relevant* information. Read "*Writing a Good Commit Message*" linked on Moodle for more guidance.

Examples of GOOD commit messages:

```
Initial commit for Assignment 2

* copied starter files over from Moodle
```

```
Completed Question 2

* full functionality complete after testing with generated files
```

```
Fixed a bug causing duplicate print messages

* extra print statements were being produce in loop
* updated logical condition to fix
```

Examples of BAD commit messages:

```
fixes
```

```
add tests
```

```
updates
```

What to Hand In

After completing and submitting your work for the Question 2, **follow steps 6 & 7 above** to create a text file named `a2-git.txt`. Be sure to include your name, NSID, student number, course number, and lecture section at the top of your `a2-git.txt` file.

Evaluation

- 5 marks: The log file shows that you used Git as part of your work for Assignment 2.

  For full marks, your log file contains
  - Meaningful commit messages.
  - A minimum of 4 separate commits.

Question 2 (25 points):

Purpose: To practice design processes, as per Chapter 5.

Degree of Difficulty: Moderate.

Restrictions: This question is homework assigned to students and will be graded. This question shall not be distributed to any person except by the instructors of CMPT 145. Solutions will be made available to students registered in CMPT 145 after the due date. There is no educational or pedagogical reason for tutors or experts outside the CMPT 145 instructional team to provide solutions to this question to a student registered in the course. Students who solicit such solutions are committing an act of Academic Misconduct, according to the University of Saskatchewan Policy on Academic Misconduct.

In this question you will apply an informal design process to the task of implementing a program at the level of Assignment 1. The purpose is to follow the process thoughtfully and reflectively. Please read the question description carefully before you start!

What to Hand In

Usually this appears last, but in this question, it's first so that you know what you have to hand in. As you can see, the code you develop is a small part of this work.

- Your plan to complete this question: `a2q2_plan.txt`
- Your design document: `a2q2_design.txt`
- Your answers to the reflection questions: `a2q2_reflections.txt`
- Your Python program `a2q2.py`

You must use **TXT** file type for this question. Be sure to include your name, NSID, student number, course number and lecture section at the top of all documents.

IMPORTANT: All of your Question 2 files should be committed to Git, see Question 1 for details!

Problem Specification

After numerous Godzilla attacks, the city of Tokyo, Japan is testing their new invention, **Mechagodzilla**. You are tasked with implementing a program to help retrieve Mechagodzilla after successful deployments.

Mechagodzilla's movements are controlled by sending a series of inputs, `'N'` for north, `'E'` for east, `'S'` for south, or `'W'` for west. If something goes wrong and Mechagodzilla is destroyed, we need to be able to find it for retrieval.

The question we want to answer: Using a standard Cartesian grid, what are the ***(x,y)* coordinates** of Mechagodzilla after it finishes a series of movement inputs? Assume Mechagodzilla always starts at the base coordinates of **(0,0)**.

Here are some examples to help clarify:

- `'NNE'`: coordinate (1, 2)
- `'NNEEENNW'`: coordinate (2, 4)
- `'WNENWNSNN'`: coordinate (-1, 4)
- `'EESSENNESE'`: coordinate (5, -1)
- `'WEENEWE'`: coordinate (2, 1)
- `'NWEWWNSWW'`: coordinate (-4, 1)

You will be given a number of datafiles containing inputs of different lengths. Each file will contain a number of lines. Each line will represent inputs from a deployment. The program you must write will open the file, read each line, and then print the final (x,y) coordinates for that deployment to the console.

For example, if a datafile contained the examples above, your program would display

```
(1, 2)
(2, 4)
(-1, 4)
(5, -1)
(2, 1)
(-4, 1)
```

Detailed Example

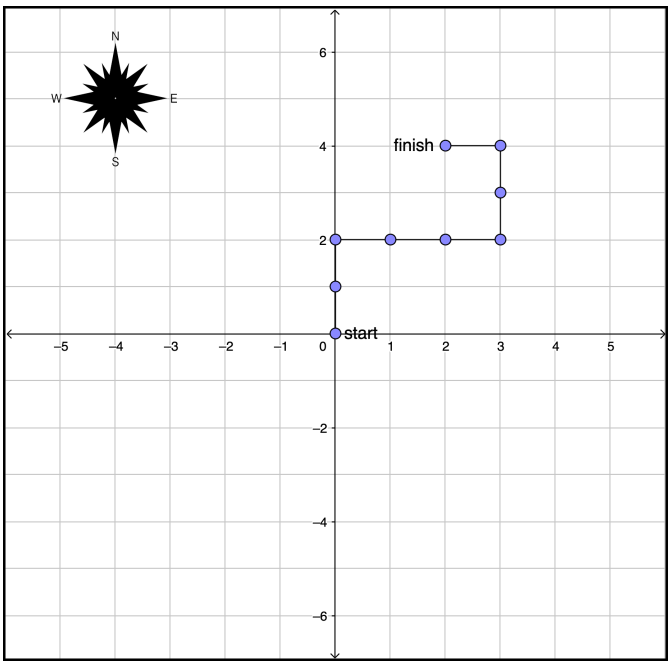
Example File (with only one deployment):

```
NNEEENNW
```

Program Output:

```
(2, 4)
```

Illustration to help visualize path taken:



Task

For this question you will do the following:

- You will start by creating a plan for your work. Your plan will schedule time for the following development phases:
  - Requirements. The specification is given above, so this part of the plan includes reading the Specification section carefully. You do not need to hand in anything for this part.
  - Design. Your plan will indicate when you will do the design phase (day, time, location), and how long you think it will take (duration).
  - Implementation. Indicate when you will work on the implementation phase (day, time, location), using your design, and how long you think it will take (duration).
  - Testing and debugging. Indicate when you will work on testing and debugging (day, time, location), and how long you think it will take (duration).
  - Final clean up for submission. Indicate when you will work on clean-up (day, time, location), and how long you think it will take (duration).

  You must submit this plan for grading.
- You will design your program, producing a design document outlining your implementation. You will submit this design for grading.
- You will record the amount of time you actually spent on each phase of the plan. You will submit this information for grading.

Details

Details about the Plan: On page 8 is an example table that you could use to outline your plan. It has columns for the information required above. You are not required to follow your plan exactly, to the last detail. It's a plan, not a commitment. It's helpful for monitoring your progress. You might find your plan didn't account for issues and problems that came up. That's okay. Make note of those in your reflections.

Details About the Design: Your design document should describe your implementation in terms of the functions you'll need to complete the program. For each function, you must specify:

- Pseudo-code or informal algorithm descriptions. The form of the description is up to you.
- An informal description of the inputs and outputs for each function.
- An informal description of any test cases you will need to check your functions.

It's important for you to keep in mind that the purpose of the design is to help you, and you should not be overly worried about the format of your design.

Details About the Implementation, Testing, and Debugging: Try to apply an incremental approach. For each function in your design:

- Implement the function according to your design.
- Test your function.
- Debug as necessary.

Keep track of your time during these phases. You'll want some objective evidence for how long things actually take, compared to what you estimated in your plan. This objective evidence will help you prepare more realistic plans in the future! How you actually complete these phases is less important than being objective about your progress.

Details About Your Submitted Program: The program is the least important part of the assignment. Make it look presentable, by adding comments and doc-strings to your functions.

Details About Your Reflection: Your answers to the reflection questions below are graded for relevance and thoughtfulness of your answer. There are no right answers, and the only way to lose these marks is to fail to reflect meaningfully.

Reflection Questions

1. How long did it take for you to create your plan?
2. Did you revise your design? In other words, did any part of your implementation turn out to be significantly different from your design? If so, briefly explain the reason for the difference(s).
3. Did any of the phases take significantly longer than you had planned? If so, explain what happened, and suggest ways to avoid this in the future.

Evaluation

- 5 marks: Your plan allocated time to every phase of the development process, and the times were plausible.
- 5 marks: Your design gave complete and useful details in terms of function descriptions, pseudo-code algorithms, and test cases.
- 5 marks: Your answers to the reflection questions were thoughtful and relevant.
- 5 marks: Your program is correct, and not terribly inefficient.
- 5 marks: Your program is well-documented with helpful comments for the reader.
- -1 mark for file formats other than TXT or PY.

Development Plan

Phase	Day	Time	Location	Duration	*Actual Time*
Requirements					
Design					
Implementation					
Testing/debugging					
Tidying up					

Question 3 (10 points):

Purpose: To reflect on the work of programming for Question 2. To practice objectively assessing the quality of the software you write. To practice visualizing improvements, without implementing them.

Degree of Difficulty: Easy.

Textbook: Chapter 4

Restrictions: This question is homework assigned to students and will be graded. This question shall not be distributed to any person except by the instructors of CMPT 145. Solutions will be made available to students registered in CMPT 145 after the due date. There is no educational or pedagogical reason for tutors or experts outside the CMPT 145 instructional team to provide solutions to this question to a student registered in the course. Students who solicit such solutions are committing an act of Academic Misconduct, according to the University of Saskatchewan Policy on Academic Misconduct.

Answer the following questions about your experience implementing the program in Question 2. You may use point form, and informal language. Just comment on your perceptions. Be brief. These are not deep questions; a couple of sentences or so ought to do it.

1. (2 marks) Comment on your program's *correctness* (see Chapter 4 of the textbook for the definition). How confident are you that your program (or the functions that you completed) is correct? What new information (in addition to your current level of testing) would raise your confidence? How likely is it that your program might be incorrect in a way you do not currently recognize?
2. (2 marks) Comment on your program's *efficiency* (see Chapter 4 of the textbook for the definition). How confident are you that your program is reasonably efficient? What facts or concepts did you use to estimate or quantify your program's efficiency?
3. (2 marks) Comment on your program's *adaptability* (see Chapter 4 of the textbook for the definition). For example, what if Question 2 asked you to write a program that had additional letters, say `'U` and `'D'`, which moved Mechagodzilla up and down (vertically)? How hard would it be to take your work in Question 2, and revise it for 3 dimensions?
4. (2 marks) Comment on your program's *robustness* (see Chapter 4 of the textbook for the definition). Can you identify places where your program might behave badly, even though you've done your best to make it correct? You do not have to fix anything you mention here; it's just good to be aware.
5. (2 marks) Consider how often you were interrupted, distracted, delayed during your work for Question 1. Do you think these factors affected substantially increased the time you needed? If so, what kinds of steps can you take to prevent these factors?

What to Hand In

Your answers to the above questions in a text file called `a2q3.pdf` (Question 3 does not need to be done using Git). Be sure to include your name, NSID, student number, course number and lecture section at the top of all documents.

Evaluation

The purpose of these questions is to reflect on your experience. You are not expected to give the "right answer", or to have worked with perfection. Your answers are for you. We will give you credit for attempting to use this opportunity to reflect in a meaningful way, no matter what your answers are.

Each answer is worth 2 marks. Full marks will be given for any answer that demonstrates thoughtful reflection. Grammar and spelling won't be graded, but practice your professional-level writing skills anyway. -1 mark for file formats other than PDF.