**Department of Computer Science**

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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

# Assignment 3
## Design, Testing and Debugging

**Date Due: Friday June 10, 11:59pm**                    **Total Marks: 70**

## 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.

- This assignment is homework assigned to students and will be graded. This assignment shall not be distributed, in whole or in part, 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 assignment 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.**

- **Assignments are being checked for plagiarism.** We are using state-of-the-art software to com-pare every pair of student submissions. Plagiarism can include: copying answers from a web page, or from a classmate, or from solutions published in previous semesters. Basically, if you cannot delete your whole assignment and do it again yourself (given adequate time), it's not your work, so don't try to claim credit for it. Your success in this course depends on what you can do, not on what you can hand in.

- 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.

- Read the purpose of each question. Read the Evaluation section of each question.

- 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 Canvas** If you are not sure, talk to a Lab TA about how to do this.

- **Canvas 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. **Do not send late assignment submissions to your instructors, lab TAs, or markers. If you require an extension, request it in advance using email.**

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

CMPT 145
Spring-Summer 2022
Principles of Computer Science

## Version History

- **06/02/2021**: A3Q1: added the requirement to submit testing. This results in an additional 5 marks for the question.

- **05/31/2021**: released to students

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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

## Question 1 (30 points):

**Five marks were added to this question for your testing. See "What to hand in" and "Evaluation".**

**Purpose:** To practice planning, designing, implementing, and testing a Python program.

**Degree of Difficulty:** Moderate. Don't leave this to the last minute. The basic functionality of the various parts of this question are easy. There are aspects of some parts that you won't appreciate until you start testing.

**References:** You may wish to review the following chapters:

- Development Processes: PCS Chapter 5
- Procedural Abstraction: PCS Chapter 6.
- Testing: PCS Chapter 7.

**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.

> **Note: A3Q2 asks about your work on A3Q1**
>
> Keep track of how long you spend working on this question, including time spent reading the question, reviewing notes and the course readings, implementing the solution, testing, and preparing to submit. Also keep track of how many times you experienced unscheduled interruptions, delays, or distractions during your work; include things like social media, phone calls, texting, but not such things as breaking for a meal, exercise, commuting to or from the university. In A3Q1 you will be asked to summarize the time you spend working on A2Q3, and your answer will be more helpful to you if you have more precise records.

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: `a3q1_plan.txt`
- Your design document: `a3q1_design.txt`
- Your answers to the reflection questions: `a3q1_reflections.txt`
- Your Python program `a3q1.py`
- **NEW!** Your test script `a3q1_testing.py`

You are allowed to use PDF, RTF, DOC files instead of TXT. Be sure to include your name, NSID, student number, course number and lecture section at the top of all documents.

**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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

## Problem Specification

> ### 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.

The program you will write will be concerned with word-chains, defined as follows. A word-chain is a sequence of words where each sequential pair of words differs by exactly one letter. For example, the following is a valid word-chain, consisting of exactly 3 words:

```
shop
stop
step
```

The words "shop" and "stop" differ by exactly one letter; the words "stop" and "step" differ by exactly one letter. Word-chains can be arbitrarily long.

However, the following is not a valid word-chain:

```
cold
coat
boat
```

It is not valid because the words "cold" and "coat" differ by more than one letter.

Your program will determine whether a word-chain is valid by the following criteria, in the following order:

1. **The word chain has at least 2 words in it.** If there are fewer than 2 words in the word-chain, your program should display the following text on the console:

   ```
   Invalid: too short
   ```

2. **Each word is a valid word.** If an invalid word (e.g., "sidx") appears in the word-chain, your program should display the following text on the console:

   ```
   Invalid word: sidx
   ```

   You only need to identify the first invalid word in the word-chain.
   To determine whether a word is valid, you will be given a file with 500 four-letter words, in alphabetical order, called `four.txt`. These words are valid; if any word in a word-chain does not appear in this file, it is not valid.

3. **No word in the word-chain appears more than once.** If a word appears more than one in the word-chain, your program should display the following text on the console:

   ```
   Invalid: repeated word: case
   ```

   You only need to identify the first repeated word in the word-chain.

4. **Each sequential pair of words differs by exactly one letter.** If two words in the word-chain differ by more than a single letter, your program should display the following text on the console:

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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

```
Invalid step from bred -> arid
```

You only need to identify the first invalid step in the word-chain.

5. **If the word-chain is valid by the above criteria,** then your program should display the following text on the console:

```
Valid!
```

Word-chain examples are provided in a ZIP file. Each example is a file containing 4-letter words, one word per line. Your program should ask for the name of a file, read the file, and check for validity. For example, it might look like this:

```
Name of chain file to check? A3Q1_09.txt
Invalid step from fell -> call
```

## Task

<div>

**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.

</div>

The purpose of this question is to get you to deliberately spend time planning, designing, implementing, and testing an application, using the concepts from Chapters 4-7. You will be expected to create a design document, along the lines of A2Q3, then implement the design, and test the implementation. To accomplish this, you should create a plan, in which you schedule the different parts of your work (see page 8).

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.

**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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

- You will record the amount of time you actually spent on each phase of the plan. You will submit this information for grading.

**Note:** If you do not engage with the planning and design, you are avoiding the main learning objective. The application you are developing is not important enough to spend time on, except to gain some experience doing the planning and design.

**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.

---

**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.

**Department of Computer Science**

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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

## 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

> **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.

- 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.

- **NEW!** 5 marks: Your test script has test cases for every function, and is plausibly thorough.

**Department of Computer Science**

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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

## Development Plan

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

## Question 2 (12 points):

**Purpose:** To reflect on the work of programming for Question 1. To practice objectively assessing the quality of the software you write. To practice visualizing improvements, without implementing improvements.

**Degree of Difficulty:** Easy.

**Textbook:** Chapter 3

**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 1. 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 3 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 3 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 3 of the textbook for the definition). For example, what if Assignment 2 asked you to write a program to check whether a $5 \times 5$ square was magic (bigger square with a larger sum, using the numbers 1 through 25)? How hard would it be to take your work in A1Q1, and revise it to handle squares of any size?

4. (2 marks) Comment on your program's *robustness* (see Chapter 3 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) How much time did you spend working on this program? Did you use the given implementation plan, or did you adjust the plan to suit your situation? Did it take longer or shorter than you planned? If anything surprised you about this task, explain why it surprised you.

6. (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? Were there any interruptions that you could not have prevented? How did you deal with those?

## What to Hand In

Your answers to the above questions in a text file called `a3q2.txt` (PDF, rtf, docx or doc are acceptable). Be sure to include your name, NSID, student number, course number and laboratory section at the top of all documents.

**Department of Computer Science**

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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

## 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.

**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

CMPT 145

Spring-Summer 2022
Principles of Computer Science

## Question 3 (28 points):

**Purpose:** To practice writing doc-strings. To practice thinking about test-case design.

**Degree of Difficulty:** Easy.

**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.

The file `a3q3.py` contains three functions, most of which are correct (but some may not be). In this question, you will be guided in the preparation of a test script to test the functions.

- `newton()`. This function implements a well-known algorithm for computing the square root of a non-negative number.
    1. Study the function, and write an appropriate doc-string for it.
    2. Design test cases for this function. There are 2 equivalence classes that are of interest.
        - Numbers between 0 and 1. The square root is bigger than the input number.
        - Numbers greater than 1. The square root is smaller than the input number.
        - There are two boundary cases: 0, and 1.
        
        Write test cases for this function, considering the information above. Don't just test integers!

- `euclid()`. This function implements a well-known algorithm for computing the greatest common divisor of two integers. For example, $euclid(27, 6) = 3$, since 3 is the largest integer that evenly divides 27 and 6.
    1. Study the function, and write an appropriate doc-string for it.
    2. Notice that the function prints to the console. It's a terrible design decision; it's nearly impossible to test console output from within Python. Before you test the function, modify the function to return the value instead.
    3. Design test cases for this function. There are several equivalence classes of interest:
        - $a > b$
        - $b > a$
        - $a = b$
        - What happens when $a = 0$ or $b = 0$? Is this okay? You may need to do some research to figure this out.
        
        Write test cases for this function, considering the information above.

**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

**CMPT 145**

Spring-Summer 2022
Principles of Computer Science

- `pascal()`. This function opens a named file and reads the contents. An example of the kind of file that it reads is given, named `triangle1.txt`. Notice that the first line in the file is the length of the example.

  1. Study the function, and write an appropriate doc-string for it.

  2. Create a test script for this function. Testing a function like this is difficult because you don't want to type in correct result in your test script. It is hard to do test functions that read from a file completely. So here is a strategy that will help:
     - The function reads "Pascal Triangles". Create a couple more example files for testing. Smaller examples are fine!
     - Test that the returned `size` is correct for the example files.
     - Test that the length of the returned `triangle` is correct.
     - Test that the elements of the triangle are also lists. To check if `alist` refers to a list, you can use the Python code `type(alist) is list`. This is a Boolean expression that returns `True` if alist was constructed by the list constructor. It will be `True` for lists, and `False` for everything else.

     Write test cases for this function, considering the information above.

- `turing()`. This function removes duplicates from a given list, keeping only the first occurrence of any duplicated value. For example, given the list `[1, 2, 3, 1]`, `turing()` will remove the second occurrence of 1.

  1. Study the function, and write an appropriate doc-string for it.

  2. Create a test script while considering the following equivalence classes:
     - No duplicates.
     - One duplicated item.
     - One duplicated item, with duplicates appearing sequentially.
     - Multiple duplicated items appearing anywhere in the list.
     - The only boundary case is the empty list.

     Write test cases for this function, considering the information above.

---

**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.

**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

CMPT 145
Spring-Summer 2022
Principles of Computer Science

## What to Hand In

- A Python script named `a3q3_newton.py` with your test script for `newton()`.

- A Python script named `a3q3_euclid.py` with your test script for `euclid()`.

- A Python script named `a3q3_pascal.py` with your test script for `pascal()`.

- A Python script named `a3q3_turing.py` with your test script for `turing()`.

- A Python script named `a3q3.py` with the functions. You've added doc-strings, and modified `euclid()` as indicated.

Be sure to include your name, NSID, student number, section number, and laboratory section at the top of all documents.

## Evaluation

> **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.

- Function interface documentation. You added the doc-strings giving the correct information.
  - 2 marks for each function.

- Your test scripts perform the indicated testing.
  - 5 marks for each function. Marks will be deducted if
    * the testing does not include sufficient test cases
    * some of the equivalence classes are not tested.
    * the test script is overly convoluted or otherwise demonstrates poor style or design