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

References, and Development Processes: Chapters 3, 4, 5

Date Due: 26 May 2021, 11:59pm **Total Marks: 57**

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 compare 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 Moodle.** If you are not sure, talk to a Lab TA about 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. **Do not send late assignment submissions to your instructors, lab TAs, or markers. If you require an extension, request it in advance using email.**

Version History

- **05/20/2021**: released to students

Question 1 (10 points):

Purpose: To work with the concept of references a bit more carefully.

Degree of Difficulty: Easy. If you understand references very well, this is not difficult.

References: You may wish to review the following chapters:

- References: CMPT 145 Readings, 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.

In class (and in the readings) we saw a version of Selection sort. As described in the readings, our implementation of selection sort works by repeatedly removing the smallest value from an unsorted list and adding it to end of a sorted list until there are no more values left in the unsorted list (see Chapter 1 of the readings).

```
1 unsorted = [3, 2, 5, 7, 6, 8, 0, 1, 2, 8, 2]
2 sorted = list()
3
4 while len(unsorted) > 0:
5     out = min(unsorted)
6     unsorted.remove(out)
7     sorted.append(out)
8
9 print(sorted)
```

One problem with this implementation is that it modifies the original list (line 6). Unless we know for sure that a list will never be needed in the future, removing all contents is a terrible idea. For small programs, you might be able to detect when this might cause problems, but in a larger script, it's almost certain that it will cause errors and bugs that are very hard to find and fix.

To address this problem, we can change the code so that a copy of the original list is made first. Since we make the copy, we can say for sure that the copy will never be needed in the future, and so removing all its contents is absolutely fine.

The file `a2q1.py` is available on Moodle, and it contains a function called `selection_sort()` which is very similar to the above code:

```
1  def selection_sort(unsorted):
2      """
3      Returns a list with the same values as unsorted,
4      but reorganized to be in increasing order.
5      :param unsorted: a list of comparable data values
6      :return:  a sorted list of the data values
7      """
8
9      result = list()
10
11     # TODO use one of the copy() functions here
12
13     while len(acopy) > 0:
14         out = min(acopy)
15         acopy.remove(out)
16         result.append(out)
17
18     return result
```

On line 11, there is a TODO item, which is where we will add code to create a copy the original unsorted list.

Also in the file are 5 different functions whose intended behaviour is to copy a list. Your job in this question is to determine which, if any, of these functions does the job right.

Note:

There is a Python list method called `copy()`, which we are not using on purpose. Our purpose here is to improve our understanding of references, which we cannot do by using a tool that evades the purpose!

Task

For each of the 5 `copy()` functions in the file `a2q1.py`, do the following:

- Determine if the function makes a copy of a list. Hint: some do not!
- If the function does create a copy, figure out how to use it in the given code. Hint: some are easy, some might take a bit of thought.
- If the function does not create a copy, explain why.
- If there is anything else going on in the function that you think might be problematic, be sure to mention it. Hint: some have this problem.

Do not change the function `selection_sort()` except to make use of one of the versions of the `copy()` function on line 11. Do not change the code for any of the `copy()` functions.

What to Hand In

Your answers to the above questions in a text file called `a2q1.txt` (PDF, rtf, docx or doc are acceptable). You might use the following format:

```
Question 1
copy1()
   - makes a copy
   - Used in the following way:
     (copy paste the lines that get it to work, not the whole function)
copy2()
   - does not make a copy
   - there is a syntax error on the third line
copy3()
   - makes a copy
   - Used in the following way:
     (copy paste the lines that get it to work, not the whole function)
   - however, it also eats all the pizza pops in the freezer
   - so don't use it!
```

The above example does not necessarily reflect the right answers!

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

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.

Evaluation

Each of the copy functions is worth 2 marks. Your answers to both questions have to be correct to get the marks.

Question 2 (10 points):

Purpose: To work with the concept of references a bit more carefully.

Degree of Difficulty: Moderate. If you understand references very well, this is not difficult.

References: You may wish to review the following chapters:

- References: CMPT 145 Readings, 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.

In the file `ascii-art.py`, you'll find a Python script that prints pictures to the console window, after opening and reading data from a compressed file. The objective of this question is to modify a couple of the functions in this script. You are invited to read the whole script, but you won't have to make changes to the trickiest parts of it.

The script reads a text file containing some data which represents an image. There are several example files given for you to try. The data files are encoded using a technique called *run-length encoding*. Once the data is read from the file, it is decoded, and turned into a 2-dimensional list of single character strings. For example:

```
[['+','+','+'],['+','+','-'],['+','-','-']]
```

Each sub-list represents a row; here we have 3 rows and three columns. We call this list of lists an *image*.

When displayed to the console, each character is displayed on a line, row by row:

```
+++
++-
+--
```

This example is not very artistic, but try out some of the examples!

There are 2 functions in the script we will be studying, and they have the following behaviour

- `flip_updown(image)`
  - Its input `image` is a list-of-lists.
  - Creates a new image containing all the rows of the original input image, but in reverse order; this corresponds to a flip across a horizontal axis.
  - Returns the new image.
  - Does not modify the original image.
- `flip_leftright(image)`
  - Its input `image` is a list-of-lists.
  - Creates a new image containing all the rows of the original input image in the same order, but each row in the new image is reversed; this corresponds to a flip across a vertical axis.
  - Returns the new image.
  - Does not modify the original image.

We will not be concerned with the other functions in the script. You can read the code, but don't be concerned if the details are a bit tricky.

Your task is to rewrite these two functions and change their behaviour to the following:

- `flip_updown(image)`
  - Its input is a list-of-lists.
  - Modifies the image so the rows are in reverse order; this corresponds to a flip across a horizontal axis.
  - Returns `None`
- `flip_leftright()`
  - Its input is a list-of-lists.
  - Modifies the image so the columns are in reverse order; this corresponds to a flip across a vertical axis.
  - Returns `None`

You'll have to adapt the script that calls these two functions (near the bottom of the file). This is part of the exercise. When you are done, your scripts produce the same images as the original script in the same order.

What to hand in:

- Hand in your working program in a file called `a2q2.py`.
- A text-document called `a2q2_demo.txt` that shows that your program working on a few examples. You may copy/paste to a text document from the console.
- If you wrote a test script, hand that in too, calling it `a2q2_testing.txt`

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.

Evaluation

- 4 marks. Your function `flip_updown(image)` correctly modifies the given image.
- 4 marks. Your function `flip_leftright(image)` correctly modifies the given image.
- 2 marks. You modified the functions doc-strings to reflect the changes you made.

Question 3 (25 points):

Purpose: To build a program and test it, starting with a design document, and an implementation plan.

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:

- General: CMPT 145 Readings, Chapter 2
- Functions Review: CMPT 141 Readings: Chapter 4
- Testing Review: CMPT 141 Readings: Chapter 15

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: A2Q4 asks about your work on A2Q3

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 A2Q4 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.

Background

A *Magic Square* is an arrangement of numbers in a square, so that every row, column, and diagonal add up to the same value. Below are two squares, but only one of them is a Magic Square.

8	1	6
3	5	7
4	9	2

1	9	6
5	3	7
4	8	2

The square on the left is a 3×3 magic square, whose rows, columns, and diagonals all sum to 15. On the right, is a 3×3 square of numbers whose rows columns and diagonals don't have the same sum.

There are magic squares of all sizes (except 2×2), but we'll be concerned with 3×3 squares, and checking if a given arrangement of 9 numbers is a magic square or not.

Definition: A 3×3 magic square is formally defined by the following three criteria:

- It contains the integers 1 through 9 inclusively.
- Every integer in the range 1 through 9 appears exactly once.
- Every row, column, and diagonal sums to 15.

In this question you will implement a program that does the following:

- It asks the user for a sequence of 9 numbers from the console. The order of the numbers is important, as the rows of the grid use this order. **For simplicity, assume that the user will type only integers on the console. For this question, you don't have worry about what to do if the user types anything other than integers.**
- The sequence of integers is converted to a list of 3 sub-lists, with each list representing a row in the grid.
- Your program checks whether the sequence of integers is a magic square or not. Your program should display the message `YES` if it's magic, or `NO` if it's not.

It's very important to point out that **you are not being asked to construct a magic square; only to check if a square is magic or not.**

Task

We've given you a design document called `MagicSquareDD.txt` (available on Moodle), which is the end result of a fairly careful design process. It describes a collection of algorithms which, when used together, will solve the problem. **You must implement the program according to the design in this document. No exceptions!** There will be some very small decisions you still have to make about the implementation. Every "Algorithm" in the design document will be a function written in Python. Function names should be very strongly similar to the given design, if not identical. Many of the Algorithms have a small number of test cases which you should use to check your implementation; you do not need to create formal test cases for Algorithms that do not have test cases given.

It's a top-down design, but the algorithms are presented in the order they should be implemented. A good rule of thumb is this: "*Design top-down. Implement bottom-up. Test each function before implementing the next.*" It's not a hard and fast rule, but it's a good guide.

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.

Development plan In Chapter 5, we stress the value of a plan for the development of software, but no example was given. The example is this question.

We have given you a table on page 11, which describes a plan to get this work done. You will notice that each algorithm in the design document has time allocated to it for implementation, and more time allocated to it for testing and debugging. Some duration estimates are given, but these do not account for your individual strengths and weaknesses, so use the plan as an example, not as an expectation. Feel free to adjust the durations given to your situation. Also, notice that we have not told you when to spend this time. Add in days and times that you will need to get this done. Don't leave it all to Wednesday.

Implementation strategy Here's how you should work with every Algorithm in the design document:

1. Read the design specification for the function.
2. Write a trivial Python function that does nothing, but has a good name, the appropriate parameters, and a return value that's appropriate. For example:

```
def check_diagonals(square):
    '''
        square: a 3x3 list of integers
        Return: True if all the diagonals sum to False otherwise
    '''
    return False
```

The function has the right parameters, and returns a value of the right kind, but doesn't really do what it is supposed to do, yet.

3. Implement the test cases for the function in a Python file named `a2q3_testing.py`. Yes, do that before you implement the function! If you don't know what we mean by test cases, review the CMPT 141 readings.
4. Run your test program. Some, if not all, of the tests should fail, because you haven't implemented the function yet. That's exactly where you want to be.
5. Implement the function carefully. Then run the test program. Debug this function as necessary until all the tests pass.
6. Go on to the next function assured that your implementation of the current function is completely correct. Do not disable the testing of any function. Every test for every function you've written should be tried every time you add a new function. That way you know if you changed something for the worse.

Because everyone is starting with the same design document, every program will have a high degree of similarity. That's perfectly okay. The value of this exercise is in the experience you gain from it, not the code you submit at then end. We don't need you to write code for us. We want you to gain experience writing code yourself.

Development Plan

Phase	**Day**	**Time**	**Duration**	*Actual Time*
Requirements			60 min	
Design	(given)		0 min	
Reading the design			20min	
Implementation				
Algorithm 7			20 min	
Algorithm 6			20 min	
Algorithm 5			20 min	
Algorithm 4			20 min	
Algorithm 3			10 min	
Algorithm 2			20 min	
Algorithm 1			10 min	
Testing/debugging				
Algorithm 7			15 min	
Algorithm 6			15 min	
Algorithm 5			15 min	
Algorithm 4			15 min	
Algorithm 3			10 min	
Algorithm 2			10 min	
Algorithm 1			10 min	
Tidying up			15 min	

What to Hand In

- Your implementation of the program: `a2q3.py`.
- Your test program `a2q3_testing.py`

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

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.

Evaluation

- 5 marks: Your program `a2q3.py` follows the design document given.
- 5 marks: Your program `a2q3.py` is well-documented with doc-strings describing parameters, and return values. The documentation standards of CMPT 141 are expected.
- 5 marks: Your test program `a2q3_testing.py` includes all the test cases given in the design document, but you are allowed to add more.
- 5 marks: Your test program `a2q3_testing.py` runs without producing any error.
- 5 marks: Your program `a2q3.py` works as described above.

Question 4 (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 5×5 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 `a2q4.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.

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.