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 9 – Solutions and Grading

Python, the UNIX Command-line, and Version Control

Date Due: Wednesday, 11 August 2021, 11:59pm **Total Marks: 25**

Overview

This assignment is a little different from past assignments. In this one, you'll be watching more video, and doing a few applications, but not so much coding as previous assignments.

This material is every bit as important as programming. Take the time to watch the videos, following along, and trying to do the same things on your computer as the video proceeds. There's a lot to watch, and none of it is very hard.

Hint: Watch all Mike's videos at 1.5x speed. You can slow down the video when you want to listen carefully. Even Mike watches Mike at 1.5x.

The questions for this assignment are simply you providing evidence that you have watched the videos, and you have given all this material a try.

The material is presented to you as follows:

- Python and the UNIX Command line
    1. Slides: Moodle, Week 13, Lab3_commandline.pdf
    2. Video: Panopto, Week 13, Python on the Command-Line
    3. Video for Windows users: Week 13, Step-by-Step Windows 10.
- Version control
    1. Slides: Moodle, Week 13, Lab04_VCS.pdf
    2. Video: Panopto, Week 13, VCS
- Note: If you prefer shorter videos, Arlen Schaeffel has created a few. These can be found in Panopto, Tutorial Material. You do not need to watch both, but you certainly may.

Don't worry that you haven't seen Lab01 and Lab02.

Mac and Linux users

You don't have to do anything special. Everything you need is built into your operating system.

Windows users

You will have to download Git-for-Windows. There is a video for that:

- Video for Windows users: Week 13, Step-by-Step Windows 10.

Do this right away. If you wait until the last minute, you will be very stressed.

Question 1 (5 points):

Purpose: Students will practice the following skills:

- Basic UNIX commands.

Degree of Difficulty: Easy. Start early, so you have time.

References: You may wish to review the following:

- Slides: Moodle, Week 13, Lab3_commandline.pdf
- Video: Panopto, Week 13, Python on the Command-Line
- Video for Windows users: Week 13, Step-by-Step Windows 10.

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.

Task

On Slides 43-55 of the Lab03 notes, there are UNIX commands and simple activities that you should do.

What to Hand In

A file called `a9q1-transcript.txt` containing the copy/paste from the work you did for Slides 43-55.

Evaluation

- 5 marks: Your copy/paste demonstrates that you carried out the activity.
  - You need to show interaction with the command-line.
  - It's okay if you typed a few things wrong, and had to correct it by trying again.

Solution: On Slides 43-55, students were required to experiment with the UNIX command-line, making use of some very basic commands, and copy/paste into a text file. Something like this:

```
Prelude[14]% pwd
/Users/mih807/Documents/Temp

Prelude[15]% ls
BINF_Hons.pdf   CMPT_Hons.pdf   Wumpus/         fun.py
CMPT145 A6/     Python/         a2q3.dot   funner.py

Prelude[16]% mkdir cmpt145

Prelude[17]% ls
BINF_Hons.pdf   CMPT_Hons.pdf   Wumpus/         cmpt145/        funner.py
CMPT145 A6/     Python/         a2q3.dot   fun.py

Prelude[18]% cd cmpt145/

Prelude[19]% pwd
/Users/mih807/Documents/Temp/cmpt145

Prelude[20]% cat > lab3file.txt
This is just some text Mike wanted us to type into a file.  It doesn't
matter what it is.  Only that we crated a new file in the new folder!

Prelude[21]% ls
labo3file.txt

Prelude[22]% more lab3file.txt

Prelude[23]% cat labo3file.txt
This is just some text Mike wanted us to type into a file.  It doesn't
matter  what it is.  Only that we crated a new file in the new folder!
```

IN the above, I added a few blank lines to make it more readable. THis was not required.

My prompt is an old UNIX tradition, long since replaced by newer traditions. In my formative years, desktop computers were rather weak, and so we were routinely logged into UNIX servers, doing our work there. So prompts usually indicated the name of the server. Also, the newest most modern shell was called `tcsh`, and the traditional symbol in the prompt was `%`; more modern shells use `$`. Finally, in the good old days, computers were named thematically. Computers I use for work are named after musical styles from the baroque period of classical music.

I used the UNIX command `cat` to create the text file. This was not required. Students were asked to use some other editor.

The command `more` is tricky to show in a transcript. Some students reported that this command was not included in their system, so may have used `less`, and that's perfectly fine. Even `cat` is fine.

Marking guidelines

- 5 marks.
- The main criteria is that the use of UNIX commands like `ls`, `pwd`, `mkdir`, `cd` was demonstrated.
- Give full marks if these commands were used in a way that shows the student knows how to use them (even if there are a few typos in the transcript).
- Give part marks if the demonstration was unconvincing in any way: too few commands, commands never used correctly.
- Give zero marks only if no transcript (text file) was submitted for this question.

Question 2 (5 points):

Purpose: Students will practice the following skills:

- Making a Python script into a command-line tool.

Degree of Difficulty: Easy. Start early, so you have time.

References: You may wish to review the following:

- Slides: Moodle, Week 13, Lab3_commandline.pdf
- Video: Panopto, Week 13, Python on the Command-Line

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.

Task

On Slide 58 of the Lab03 notes, do Activity 1.

What to Hand In

A file called `a9q2-transcript.txt` containing the copy/paste from the activity. Please make sure that this part of your file is clearly marked as "Activity 1". This is so the markers can find it easily.

Evaluation

- 5 marks: Your copy/paste demonstrates that you carried out the activity.
  - You need to show interaction with the command-line and at least 3 different uses of the script taking different values from the command-line.

Solution: Students were instructed to download `fact.py`, and modify it so that it behaves like a previous example. Then they were to run the new version on the command-line three times.

A program that uses command-line arguments appears in the file `fact2.py`. Running it 3 times would look like this:

```
Activity 1
==========

Prelude[3]% python fact.py
Traceback (most recent call last):
  File "fact.py", line 33, in <module>
    example = int(sys.argv[1])
IndexError: list index out of range
Prelude[4]% python fact.py 5
120
Prelude[5]% python fact.py 5
120
Prelude[6]% python fact.py 2
2
Prelude[7]% python fact.py 0
1
```

This version of the program does not have to handle an incorrect number of command-line arguments.

Marking guidelines

- 5 marks.
- There will be variations in prompts. On Windows (git-bash), the prompt actually extends over two lines.
- The transcript has to show that python was called on the command-line with the `fact.py` script, and a command-line argument.
- Deduct 1 mark if there are fewer than 3 different examples.
- Deduct 3 marks if the output (factorial) is wrong!
- Give 0 (zero) marks if the command-line is not being used.

Question 3 (5 points):

Purpose: Students will practice the following skills:

- Making a Python script into a command-line tool.

Degree of Difficulty: Easy. Start early, so you have time.

References: You may wish to review the following:

- Slides: Moodle, Week 13, Lab3_commandline.pdf
- Video: Panopto, Week 13, Python on the Command-Line

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.

Task

On Slide 59 of the Lab03 notes, do Activity 2. This calls for a minor revision to the factorial script. Run the script a couple of times, using command-line arguments, and at least one time with no arguments, showing your helpful message.

What to Hand In

A file called `a9q3-transcript.txt` containing the copy/paste from the activity. Please make sure that this part of your file is clearly marked as "Activity 2". This is so the markers can find it easily.

Evaluation

- 5 marks: Your copy/paste demonstrates that you carried out the activity.
  - You need to show interaction with the command-line and at least 3 different uses of the script taking different values from the command-line.
  - You demonstrated that the new version of your script will display a helpful message if the script is called without the right number of command-line arguments.

Solution: This is the same script as Question 2, but this time, it should handle the case where an incorrect number of command-line arguments is used.

A program that uses command-line arguments appears in the file `fact2.py`. Running it 3 times would look like this:

```
Activity 2
==========

Prelude[11]% python fact.py
Usage: fact.py <n>
Prelude[12]% python fact.py 2
2
Prelude[13]% python fact.py 7
5040
Prelude[14]% python fact.py 7 7 7
Usage: fact.py <n>
```

Marking guidelines

- 5 marks.
- There will be variations in prompts. On Windows (git-bash), the prompt actually extends over two lines.
- The command-line has to use python, the `fact.py` script, and a command-line.
- Deduct 3 marks if the transcript does not demonstrate a helpful message if the wrong number of arguments is given. The content of that line is irrelevant.
- Deduct 1 mark if using the script with the wrong number of command-line arguments still results in a runtime error.
- Give 0 (zero) marks if the command-line is not being used.

Question 4 (5 points):

Purpose: Students will practice the following skills:

- Making a Python script into a command-line tool.

Degree of Difficulty: Easy. Start early, so you have time.

References: You may wish to review the following:

- Slides: Moodle, Week 13, Lab3_commandline.pdf
- Video: Panopto, Week 13, Python on the Command-Line

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.

Task

On Slide 60-62 of the Lab03 notes, do Activity 3. This calls for a minor revision to a script from Chapter 3.

What to Hand In

A file called `a9q4-transcript.txt` containing the copy/paste from the activity.

Evaluation

- 5 marks: Your copy/paste demonstrates that you carried out the activity.
  - Your copy/paste shows the script running on the command-line before you changed it.
  - Your copy/paste shows that you revised the script to take 2 command-line arguments, and you have called the script a number of times.
  - You demonstrated that you can find a value for n so that the output is around 40%-60% dead ends. You don't need to be really precise. Just show that you can run the script a bunch of times, without having to edit the script to change n and $trials$ every time.

Solution: Students were instructed to download `self-avoiding-random-walk.py`, and modify it so that it takes 2 command-line arguments: n and $trials$. Then they were to run the new version on the command-line until they could find outputs in the range 40%-60% consistently.

A program that uses command-line arguments appears in the file `self-avoiding-random-walk.py`. The transcript should show something like this:

```
Activity 3
==========

Prelude[30]% python self-avoiding-random-walk.py 10
Usage: self-avoiding-random-walk.py <n> <trials>
Prelude[31]% python self-avoiding-random-walk.py 10 10
0.0% dead ends
Prelude[32]% python self-avoiding-random-walk.py 100 10
100.0% dead ends
Prelude[33]% python self-avoiding-random-walk.py 60 10
90.0% dead ends
Prelude[34]% python self-avoiding-random-walk.py 60 10
100.0% dead ends
Prelude[35]% python self-avoiding-random-walk.py 60 10
80.0% dead ends
Prelude[36]% python self-avoiding-random-walk.py 35 10
50.0% dead ends
Prelude[37]% python self-avoiding-random-walk.py 35 10
60.0% dead ends
Prelude[38]% python self-avoiding-random-walk.py 35 10
60.0% dead ends
Prelude[39]% python self-avoiding-random-walk.py 35 10
70.0% dead ends
Prelude[40]% python self-avoiding-random-walk.py 35 10
60.0% dead ends
Prelude[41]% python self-avoiding-random-walk.py 35 100
72.0% dead ends
Prelude[42]% python self-avoiding-random-walk.py 35 100
65.0% dead ends
Prelude[43]% python self-avoiding-random-walk.py 35 100
73.0% dead ends
Prelude[44]% python self-avoiding-random-walk.py 22 100
34.0% dead ends
Prelude[45]% python self-avoiding-random-walk.py 22 100
42.0% dead ends
Prelude[46]% python self-avoiding-random-walk.py 22 100
36.0% dead ends
Prelude[47]% python self-avoiding-random-walk.py 22 100
36.0% dead ends
Prelude[48]% python self-avoiding-random-walk.py 28 100
44.0% dead ends
Prelude[49]% python self-avoiding-random-walk.py 28 100
52.0% dead ends
Prelude[50]% python self-avoiding-random-walk.py 28 100
55.0% dead ends
Prelude[51]% python self-avoiding-random-walk.py 28 100
61.0% dead ends
```

```
Prelude[52]% python self-avoiding-random-walk.py 28 100
56.0% dead ends
```

Students should note how I used binary search to locate a value of n. Binary search is not just for lists and trees!

Marking guidelines

- 5 marks.
- The values for n and $trials$ should not figure into the grading. The exercise was motivation to run the script repeatedly with different command-line arguments.
- Give full marks for any demonstration of the script being used on the command-line to experiment with the inputs.
- Deduct 1 mark if there are fewer than 5 attempts.
- Give 0 (zero) marks if the command-line is not being used.

Question 5 (5 points):

Purpose: Students will practice the following skills:

- Using Git version control for a simple project.

Degree of Difficulty: Easy. Start early, so you have time.

References: You may wish to review the following:

- Slides: Moodle, Week 13, Lab04_VCS.pdf
- Video: Panopto, Week 13, VCS
- Note: If you prefer shorter videos, Arlen Schaeffel has created a few. In order:
  - Panopto, Tutorial Material: Version Control Overview
  - Panopto, Tutorial Material: Version Control Basics
  - Panopto, Tutorial Material: Version Control – Command Line
  - Panopto, Tutorial Material: Version Control – GUI
  - Panopto, Tutorial Material: Version Control – Branching and Merging

  The total time is about the same, and the content is more or less the same, but you might prefer his presentation. Watching at 1.5x is recommended.

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.

Task

On Slide 24-45 of the Lab04 notes, do Activity 3. This is basically a chance for you to repeat the steps shown on video, in a step by step manner. Take time to pause and work things out.

What to Hand In

On Slide 45, you are instructed to submit a log-file. In PyCharm, or on the command-line, type the command `git --no-pager log`. This will output text to the console window, and it may scroll past the limits of your window. Copy and paste the information displayed into a text file called `a9q5-log.txt`.

Evaluation

- 5 marks: Your copy/paste demonstrates that you carried out the activity.
  - Your log shows multiple commits; three commits is the minimum.
  - Your log shows good commit messages.

Solution: Students were asked to initialize a local Git repo in a PyCharm project, add a script, make a few edits, and and a few commits. THe video gave a walk-through, and the lab slides reviewed the steps required.

Here's a plausible log-file:

```
Prelude[44]% git --no-pager log
commit 93b5f1248c5f4bc13c86b23eac42845b6aec0e96 (HEAD -> master)
Author: Michael Horsch <horsch@cs.usask.ca>
Date:   Wed Aug 11 09:47:03 2021 -0600

    Replaced recursion with a loop.

commit eee374dca007b994416b68423ae74b4bd62317f0
Author: Michael Horsch <horsch@cs.usask.ca>
Date:   Wed Aug 11 09:44:44 2021 -0600

    Implemented recursive factorial.

commit 99cdc9cf6aa1c817b09591d156f23beb5934463d
Author: Michael Horsch <horsch@cs.usask.ca>
Date:   Wed Aug 11 09:42:55 2021 -0600

    Added stub function for factorial.

commit 1fb60fced17669384695020a24aa55bec60e8bc0
Author: Michael Horsch <horsch@cs.usask.ca>
Date:   Wed Aug 11 09:41:31 2021 -0600

    Initialized with a silly file from the previous lab.
```

Marking guidelines

- 5 marks. For full marks:
  - There have to be at least 3 commits in the log-file.
  - The commit messages have to indicate that work was done. Unacceptable:
    - Single word messages.
    - Messages that do not describe the work done.