CMPUT274—Assignment 1 (Fall 2025)

R. Hackman

Due Date: Friday September 26th, 8:00PM

Per course policy you are allowed to engage in *reasonable* collaboration with your classmates. You must include in the comments of your assignment solutions a list of any students you collaborated with on that particular question.

For each assignment you may not use any Python features or functions not discussed in class! This means, for example, you may not use loops, casting, any built-in functions or methods we have not discussed. If you are in doubt, you may ask on the discussion forum — however remember if you are including your code, or even your plan to solve a question, you must make a private post.

Unless explicitly forbidden by a question, you are always allowed to write additional helper functions that may help you solve a problem. In fact in some cases it will be *necessary* to write helper functions!

1. **This question is about downloading the course repository, you do not have to write any code for this question. However, the files required for all future questions are included in this repository, without completing this question you cannot work on the subsequent questions.** The course repository is a Git repository hosted on GitHub where all files related to the course will be distributed. This is where all future course material will be distributed from. This means assignments, lecture slides, lecture code examples, and more will all be distributed via the git repository and *not* through the canvas page. As such it is *very* important that you do complete this question.

   For this question you only have to clone the course repository, so that you are able to access these files. This means you must install Git. The instructions below are for the two course-supported options. If a student wishes to work with some other tool that is up to them, but they do so at their own risk and are not guaranteed help from course staff in solving any issues that arise.

   For Mac: Follow the steps below

   (a) Run the command below, and follow the instructions it provides to you

   ```
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
   ```

   (b) After completing the previous step, run the following command

   ```
   brew install git
   ```

   For WSL on Windows: Run the following command in your WSL shell:

   ```
   sudo apt install git -y
   ```

For all, after previous steps complete: Once you have installed Git you need to run the command to clone the repository. To keep things simple for future commands, we request that you clone the repository from your home directory. To clone the repository in your home directory run the following two commands:

```
cd
git clone https://github.com/rsh-ua/F25-CMPUT274.git
```

Which will create a directory for you named `F25-CMPUT274` in your current working directory, with all the currently uploaded files in it. In the future whenever you attend class or work on this course you should start by entering that directory and running the command **git pull** in order to pull all the newly added files and updates to the repository.

Do not edit files in this directory! Use this directory as read only, and if you want to use files from it make a copy of them in another location for editing.

Once you have cloned the repository, you must install the CMPUT274 Python module that will be used in this course. The following steps are to install the module. The module installation is very simplistic, and requires you do not change or rename/move any of the directories that make up the repository after installation is complete.

To install the CMPUT274 Python module, navigate to the `module` directory in the repository. If you cloned the repository in a location other than your home directory, you must navigate to the module directory yourself. Assuming you followed the steps above correct this can be done with the command:

```
cd ~/F25-CMPUT274/module
```

Once you have navigated to the `module` directory run the following command:

```
  ./install.sh
```

Running that command should print a message that the installation is complete. If anything else prints then read the error message and do as it says. If you run that command again you should receive a message stating that the installation has already been completed.

Lastly, you must test the installation of the module. To do so follow the steps below:

(a) Close the shell you used to install the module.

(b) Open up a new shell (your wsl ubuntu instance on Windows, or Terminal on Mac)

(c) Run your Python interpreter and run the following statements
   - `from cmput274 import *`
   - `a1Code()`

(d) Copy the result of the function call to `a1Code` into a text file named `a1q1.txt` to be included in your submission per the submission instructions at the end of the assignment.

Deliverables: For this question include in your final submission zip the plaintext file `a1q1.txt`

2. **Writing Testcases** — in this question you will be writing test cases for some functions that are already provided, and learn about the testing functions provided in the `cmput274` Python module.

   Find the file `q2.py` in the provided files of the assignment in the course repository. Make your own copy of the file outside the course repository. Write your test cases in the copy you made of `q2.py`. If you are struggling with how to do this, you can follow the steps below assuming you followed all the instructions for cloning the repository above.

   ```
   cd
   mkdir myA1
   cd myA1
   cp ~/F25-CMPUT274/a1/provided/q2.py ./
   ```

   In order to write test cases you will be calling two of the functions provided to you in the `cmput274` module. These functions are `testWithin` and `testExact`. These are described both in the documentation of the module and the `q2.py` file. In order to access the help information of the `cmput274` module you can do the following:

   - In a Python shell run the command `import cmput274`
   - In the same Python shell run the command `help(cmput274)`
   - This provides help documentation on everything currently provided by the module, including the `testWithin` and `testExact` functions.
   - For just help on these functions, if you have already run the command `from cmput274 import *`, then you can simply say `help(testWithin)` or `help(testExact)`

   Follow the instructions in `q2.py` to write test cases to test the functions `f1` and `f2`. Place your test cases in the `main` function as specified in the instructions in the file. You may run the test cases that are written in that file at any time by running the command `python3 q2.py` so long as your shell is in the current working directory that contains your copy of `q2.py`.

   Deliverables: For this question include in your final submission zip the python file `q2.py`

3. In this question you will write a Python function `charCount` which takes two string parameters, the first of which is a single-character string and the second is any string at all. Your function returns an integer, which is the number of times the given character occurs in the other string parameter. For example

   ```
   charCount("l", "hello my friendly pal") -> 4
   charCount("x", "abcd1234") -> 0
   ```

   You are again provided a skeleton file `q3.py` to fill in. This provides some basic test cases already written for you, as well as empty function definition. While there are two basic test cases provided, they are insufficient to effectively test your function. In addition to filling in the function definition you should write your own test cases to be confident your solution works.

   Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. If you have followed all instructions in the previous questions so far, then you can make a copy of this file by running the commands below:

   ```
   cd ~/myA1
   cp ~/F25-CMPUT274/a1/provided/q3.py ./
   ```

   Deliverables: For this question include in your final submission zip the python file `q3.py`

4. In this question you will write a Python function `isPrime` which takes a single integer parameter and returns `True` if it is a prime number and `False` otherwise. For example

   ```
   isPrime(7) -> True
   isPrime(91) -> False
   ```

   Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. If you have followed all instructions in the previous questions so far, then you can make a copy of this file by running the commands below:

   ```
   cd ~/myA1
   cp ~/F25-CMPUT274/a1/provided/q4.py ./
   ```

   Deliverables: For this question include in your final submission zip the python file `q4.py`

5. In this question you will write a Python function `digitSkip` which converts an integer number into the resultant integer number by applying our *digit skip* transformation. The digit skip transformation is described as follows

   - The "current digit" begins at the least-significant digit
   - The current digit becomes the next most-significant digit of your result. This can be thought of as the current digit being placed *left* of your result so far.
   - The current digit is updated to be the next digit in the original number that is a number of places to the left of the current digit equal to the current digit plus one. This can be thought of as the current digit telling you how many digits to *skip* in the original number. For example, if the current digit is 0 then you would just go to the next digit over, skipping nothing. If the current digit is 2 then you would skip over the next two digits, going to the third digit to the left of the 2 in your original number
   - When all digits are moved past, the result is finished.

For example, consider this procedure applied to the number 139210014.

Current digit, coloured red	Result so far
139210014	4
139210014	24
139210014	124

Some examples of the `digitSkip` function are

```
digitSkip(139210014) -> 124
digitSkip(51001) -> 101
digitSkip(513849) -> 9
```

Once again, you should not edit any files in the repository itself, but instead make your own copy of it to edit. If you have followed all instructions in the previous questions so far, then you can make a copy of this file by running the commands below:

```
cd ~/myA1
cp ~/F25-CMPUT274/a1/provided/q5.py ./
```

Deliverables: For this question include in your final submission zip the python file `q5.py`

How to submit: Create a zip file `a1.zip`, make sure that zip file contains your created text file `a1q1.txt`, four python programs `q2.py`, `q3.py`, `q4.py`, and `q5.py`. Assuming all four of these files are in your current working directory you can create your zip file with the command

```
$ zip a1.zip a1q1.txt q2.py q3.py q4.py q5.py
```

Upload your file `a1.zip` to the a1 submission link on Canvas.