Week 06 Weekly Test Questions

Test Conditions

These questions must be completed under self-administered exam-like conditions. You must time the test yourself and ensure you comply with the conditions below.

- You may complete this test in CSE labs or elsewhere using your own machine.
- You may complete this test at any time before **Week 7 Thursday 21:00:00**.
- Weekly tests are designed to act like a past paper - to give you an idea of how well you are progressing in the course, and what you need to work on. Many of the questions in weekly tests are from past final exams.
- Once the first hour has finished, you must submit all questions you've worked on.
- You should then take note of how far you got, which parts you didn't understand.
- You may choose then to keep working and submit test question anytime up to Week 7 Thursday 21:00:00
- However the maximum mark for any question you submit after the first hour will be 50%

You may access this **language documentation** while attempting this test:

- manual entries, via the *man* command.
- Texinfo pages, via the *info* command.
- Bash documentation, via the `help` build-in.
- Python documentation, via the `python3 -c 'help()'` command.
- Shell/Regex quick reference
- Python quick reference
- full Python 3.9 documentation

Any violation of the test conditions will results in a mark of zero for the entire weekly test component.

Getting Started

Set up for the test by creating a new directory called `test06` and changing to this directory.

```
$ mkdir test06
$ cd test06
```

There are no provided files for this test.

Test Complete!

Your time for this test has finished. You must submit your work now. You should reflect on how you went in this hour, and discuss with your tutor if you have concerns. You may choose to keep working, but the maximum mark for any questions you submit later will be 50%.

WEEKLY TEST QUESTION:

Create A File of Integers In Python

Write a Python program, `create_integers_file.py` which takes 3 arguments.

The first and second arguments will specify a range of integers.

The third argument will specify a filename.

Your program should create a file of this name containing the specified integers.

For example:

```
$ ./create_integers_file.py 40 42 fortytwo.txt
$ cat fortytwo.txt
40
41
42
$ ./create_integers_file.py -5 5 a.txt
$ cat a.txt
-5
-4
-3
-2
-1
0
1
2
3
4
5
$ ./create_integers_file.py 1 1000 a.txt
$ wc a.txt
1000 1000 3893 a.txt
```

NOTE:

You assume your program is give three valid arguments.

Your program can assume its first and second arguments will be integers.

Your program can assume its second argument is greater than or equal to its first argument.

Your answer must be Python only. You can not use other languages such as Shell, Perl or C.

Make the first line of your script `#!/usr/bin/env python3`

You may not run external programs. For example, you can't run `seq` .

No error checking is necessary.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 2041 autotest python_create_integers_file
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 test06_python_create_integers_file create_integers_file.py
```

WEEKLY TEST QUESTION:

Print the N-th Line of a File

Write a Python program, `nth_line.py` which prints the *n*-th line of a file.

It will be given two arguments *n* and the file name.

Your program should print nothing if the file does not have an *n*-th line

```
$ seq 42 99 > numbers.txt
$ head numbers.txt
42
43
44
45
46
47
48
49
50
51
$ tail numbers.txt
90
91
92
93
94
95
96
97
98
99
$ ./nth_line.py 1 numbers.txt
42
$ ./nth_line.py 20 numbers.txt
61
$ ./nth_line.py 1000 numbers.txt
$ echo this file has one line > file.txt
$ cat file.txt
this file has one line
$ ./nth_line.py 1 file.txt
this file has one line
$ ./nth_line.py 42 file.txt
```

NOTE:

You assume your program is give two valid arguments.

Your program can assume its first argument will be a positive integer.

You should not assume anything about the lines in the file.

You can assume the file contains ASCII text.

Your answer must be Python only. You can not use other languages such as Shell, Perl or C.

Make the first line of your script `#!/usr/bin/env python3`

You may not run external programs. For example, you can't run `sed` .

No error checking is necessary.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 2041 autotest python_nth_line
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 test06_python_nth_line nth_line.py
```

WEEKLY TEST QUESTION:

Reverse the Lines of a File into a New File

Write a Python program, `reverse_file.py` which creates a file that is the reverse of another files.

It will be given two arguments the file name of the source file, and the file name of the destination file.

Your program should print nothing.

```
$ seq 42 99 > numbers.txt
$ head numbers.txt
42
43
44
45
46
47
48
49
50
51
$ tail numbers.txt
90
91
92
93
94
95
96
97
98
99
$ ./reverse_file.py numbers.txt new_numbers.txt
$ head new_numbers.txt
99
98
97
96
95
94
93
92
91
90
$ tail new_numbers.txt
51
50
49
48
47
46
45
44
43
42
$ echo this file has one line > file.txt
$ cat file.txt
this file has one line
$ ./reverse_file.py file.txt reverse.txt
$ cat reverse.txt
this file has one line
```

> NOTE:
>
> You assume your program is give two valid arguments.
>
> Your program can assume the two arguments are not the same file.
>
> You can assume the file contains ASCII text.
>
> Your answer must be Python only. You can not use other languages such as Shell, Perl or C.
>
> Make the first line of your script `#!/usr/bin/env python3`
>
> You may not run external programs. For example, you can't run `tac` .
>
> No error checking is necessary.

When you think your program is working you can `autotest` to run some simple automated tests:

```
$ 2041 autotest python_reverse_file
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs2041 test06_python_reverse_file reverse_file.py
```

Submission

When you are finished each exercise make sure you submit your work by running **give**.

You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Week 7 Thursday 21:00:00** to complete this test.

Automarking will be run by the lecturer several days after the submission deadline for the test, using test cases that you haven't seen: different to the test cases `autotest` runs for you.

Hint: do your own testing as well as running `autotest`

Test Marks

After automarking is run by the lecturer you can view it here the resulting mark will also be available via via give's web interface or by running this command on a CSE machine:

```
$ 2041 classrun -sturec
```

The test exercises for each week are worth in total 1 marks.

Each test is worth 1.7 marks, and will be automarked. Your total mark for the tests component is computed as a sum of your best 6 of 8 test marks.
