Week 08 Weekly Test Sample Answers

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 9 Saturday 12:00:00**.
- 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 9 Saturday 12:00:00
- 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 `test08` and changing to this directory.

```
$ mkdir test08
$ cd test08
```

There are some provided files for this test which you can fetch with this command:

```
$ 2041 fetch test08
```

If you're not working at CSE, you can download the provided files as a zip file or a tar file.

WEEKLY TEST QUESTION:

Echo If Consecutive Three Vowels

Write a Python program **three_vowel_echo.py** that prints its command-line argument to standard output, similar to **echo** command in Shell, except arguments should be printed only if they contain 3 consecutive vowels.

```
$ ./three_vowel_echo.py Pompeii Rome Aeolian Florence
Pompeii Aeolian
$ ./three_vowel_echo.py

$ ./three_vowel_echo.py an anxious bedouin beauty booed an ancient zoologist
anxious bedouin beauty booed
$ ./three_vowel_echo.py abstemiously adenocarcinomatous Hawaiian Eoanthropus
abstemiously Hawaiian Eoanthropus
```

NOTE:

Your program can assume vowels are there are 5 vowel ['a', 'e', 'i', 'o', 'u']
and their upper-case equivalents ['A', 'E', 'I', 'O', 'U']

You can assume input lines are ASCII, you should not assume anything else about input lines.

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

You may not run external programs.

No error checking is necessary.

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

```
$ 2041 autotest three_vowel_echo
```

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

```
$ give cs2041 test08_three_vowel_echo three_vowel_echo.py
```

SOLUTION:

Sample solution for `three_vowel_echo.py`

```
#!/usr/bin/env python3


"""
print command line arguments which contain 3 consecutive vowels
written by d.brotherston@unsw.edu.au for COMP2041/9044
"""


from sys import argv
from re import search, I

words = []

def main():
    for arg in argv[1:]:
        if search(r'[aeiou]{3}', arg, flags=I):
            if arg[-1] == '\n': arg = arg[:-1]
            words.append(arg)

    print(' '.join(words))

if __name__ == "__main__":
    main()
```

SOLUTION:

Alternative solution for `three_vowel_echo.py`

```
#!/usr/bin/env python3


"""
print command line arguments which contain 3 consecutive vowels
written by d.brotherston@unsw.edu.au for COMP2041/9044
"""


import sys, re

print(" ".join(filter(lambda x: re.search(r'[aeiou]{3}', x, flags=re.I), sys.argv[1:])))
```

WEEKLY TEST QUESTION:

Print the Middle Lines of A File

Write a Python program **middle_lines.py** which prints the middle line(s) in a file.

If the file contains an odd number of lines it should print one line.

If the file contains an even number of lines it should print two lines.

If the file contains no lines it should print nothing.

```
$ cat odd.txt
line 0
line 1
line 2
line 3
line 4
$ ./middle_lines.py odd.txt
line 2
$ cat even.txt
line 0
line 1
line 2
line 3
line 4
line 5
$ ./middle_lines.py even.txt
line 2
line 3
$ ./middle_lines.py /dev/null
```

NOTE:

You can assume one and only one file is given as argument and that it exists and it is readable.

You can assume lines in the file are terminated with a '\n' byte, and the file contains no un-terminated lines.

You can read the entire file into memory.

You can assume the file contains only ASCII bytes.

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

You may not run external programs.

No error checking is necessary.

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

```
$ 2041 autotest middle_lines
```

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

```
$ give cs2041 test08_middle_lines middle_lines.py
```

SOLUTION:

Sample solution for `middle_lines.py`

```python
#!/usr/bin/env python3
"""
print middle line(s) of a file
written by d.brotherston@unsw.edu.au for COMP2041/9044
"""

from sys import argv

def main():
    with open(argv[1], "r") as f:
        lines = f.readlines()
        n_lines = len(lines)
        if n_lines == 0:
            # empty file
            pass
        elif n_lines % 2 == 1:
            # odd number of lines
            print(lines[n_lines // 2], end="")
        else:
            # even number of lines
            print(lines[(n_lines // 2) - 1], end="")
            print(lines[(n_lines // 2)], end="")

if __name__ == "__main__":
    main()
```

WEEKLY TEST QUESTION:

Print the lines of A File Sorted on Length

Write a Python program, `sort_file_lines.py` which is given one argument, a file name.

Your program should print the lines of the file in order of length, shortest to longest.

Lines of equal length should be sorted alphabetically.

```
$ cat file.txt
tiny
short line
medium line
longgggggg line
a equal line
b equal line
c equal line
even longgggggggggggggggggggggggggggggggggggggggggggerr
$ sort_file_lines.py file.txt
tiny
short line
medium line
a equal line
b equal line
c equal line
longgggggg line
even longgggggggggggggggggggggggggggggggggggggggggggerr
```

NOTE:

You can assume one and only one file is given as argument and that it exists and it is readable.

You can assume lines in the file are terminated with a '\n' byte, and the file contains no un-terminated lines.

You can assume the file contains only ASCII bytes.

You can read the entire file into memory.

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

You may not run external programs.

No error checking is necessary.

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

```
$ 2041 autotest sort_file_lines
```

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

```
$ give cs2041 test08_sort_file_lines sort_file_lines.py
```

SOLUTION:

Sample solution for `sort_file_lines.py`

```
#!/usr/bin/env python3
"""
print the lines of a files sorted on length, shortest to longest
if lines are of equal length they sorted alphabetically
written by d.brotherston@unsw.edu.au for COMP2041/9044
"""

from sys import argv

def main():
    with open(argv[1], "r") as f:
        lines = f.readlines()
        lines.sort() # first sort alphabetically
        lines.sort(key=len) # then sort by length
        print("".join(lines), end="")

if __name__ == "__main__":
    main()
```

SOLUTION:

Alternative solution for `sort_file_lines.py`

```
#!/usr/bin/env python3
"""
print the lines of a files sorted on length, shortest to longest
if lines are of equal length they sorted alphabetically
written by d.brotherston@unsw.edu.au for COMP2041/9044
"""

from sys import argv
from functools import cmp_to_key

def compare(line1, line2):
    if len(line1) != len(line2):
        return len(line1) - len(line2)
    return line1 < line2

def main():
    with open(argv[1], "r") as f:
        lines = f.readlines()
        lines = sorted(lines, key=cmp_to_key(compare))
        print("".join(lines), end="")

if __name__ == "__main__":
    main()
```

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 9 Saturday 12:00:00** 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.

The best 6 of your 8 test marks for weeks 3-10 will be summed to give you a mark out of 9.
