Week 07 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 8 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 8 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 `test07` and changing to this directory.

```
$ mkdir test07
$ cd test07
```

There are no provided files for this test.

WEEKLY TEST QUESTION:
Unique Echo

Write a Python program **uniq_echo.py** that prints its command-line argument to standard output, similar to **echo** command in Shell, except only the first occurrence of any argument should be printed, Repeat occurrences should not be printed.

```
$ ./uniq_echo.py echo echo echo
echo
$ ./uniq_echo.py bird cow bird cow fish bird cow fish bird
bird cow fish
$ ./uniq_echo.py how much wood would a woodchuck chuck
how much wood would a woodchuck chuck
$ ./uniq_echo.py d c b d c a a d
d c b a
$ ./uniq_echo.py
```

NOTE:

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

You may not run external programs.

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

```
$ 2041 autotest uniq_echo
```

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

```
$ give cs2041 test07_uniq_echo uniq_echo.py
```

SOLUTION:

Sample solution for `uniq_echo.py`

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

# print command line arguments unless they are a repeat
# written by d.brotherston@unsw.edu.au for COMP2041/9044

import sys

seen = set()
for arg in sys.argv[1:]:
    if arg not in seen:
        print(arg, end=" ")
        seen.add(arg)
print()
```

SOLUTION:

Alternative solution for `uniq_echo.py`

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

# print command line arguments unless they are a repeat
# written by d.brotherston@unsw.edu.au for COMP2041/9044

import sys
from collections import OrderedDict

print(*list(OrderedDict.fromkeys(sys.argv[1:])))
```

WEEKLY TEST QUESTION:

Snap N

Write a Python program **snap_n.py** which reads lines from its input until it reads an line that has been entered *n* times.

Your program should then print "Snap: " followed by the line.

Your program will be given *n* as a command line argument. Your program should print nothing if the end-of-input is reached before any line is repeated *n* times.

```
$ ./snap_n.py 2
hi
how
are
you
hi
Snap: hi
```

```
$ ./snap_n.py 2
hi
hi hi
hi hi hi
hi hi hi hi
hi hi
Snap: hi hi
```

```
$ ./snap_n.py 4
Hello World
Line 2
Hello World
Line 3
Hello World
Line 4
Hello World
Snap: Hello World
```

NOTE:

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 snap_n
```

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

```
$ give cs2041 test07_snap_n snap_n.py
```

SOLUTION:

Sample solution for `snap_n.py`

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

# Stop after a line on STDIN is seen n times
# n is supplied as a command line argument
# written by d.brotherston@unsw.edu.au for COMP(2041|9044)

import sys
from collections import Counter

n = int(sys.argv[1])

counter = Counter()
for line in sys.stdin:
    line = line.rstrip("\n")
    counter[line] += 1
    if counter[line] == n:
        print("Snap:", line)
        break
```

WEEKLY TEST QUESTION:
Print the Line(s) from Stdin With the Largest Number

Write a Python program **largest_numbered_line.py** that read lines from standard input until end-of-input. It should then print the line(s) which contained the largest number.

Lines may contain multiple numbers and they may contain any number of any character between the numbers.

If the largest number occurs on multiple lines you should print all of the lines in the order they occurred.

If no line contains a number, your program should print nothing.

```
$ ./largest_numbered_line.py
I spent $ 15.50 for
3.3 kg apples yesterday.
2000 is a leap year.
Ctrl-D
2000 is a leap year.
```

```
$ ./largest_numbered_line.py
two2 four4 eight8 sixteen16
1 sixteen-and-half 16.5  1
11 12 13
Ctrl-D
1 sixteen-and-half 16.5  1
```

```
$ ./largest_numbered_line.py
the quick brown f42ox
4 9 42 2 4
1 2 3 4 42.0
no forty two
last 42
Ctrl-D
the quick brown f42ox
4 9 42 2 4
1 2 3 4 42.0
last 42
```

```
$ ./largest_numbered_line.py
a 0.01
b .5
c -0.9
Ctrl-D
b .5
```

```
$ ./largest_numbered_line.py
a -.5
b -5
c --90--
Ctrl-D
a -.5
```

```
$ ./largest_numbered_line.py
I love programming in Shell.
but I like Python better.
Ctrl-D
```

NOTE:

You can assume numbers do not contain white space, commas or other extra characters.

You can assume numbers are only in **decimal format**.

You can assume numbers are not in scientific/exponential format.

You can assume input is only ASCII.

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 largest_numbered_line
```

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

```
$ give cs2041 test07_largest_numbered_line largest_numbered_line.py
```

SOLUTION:

Sample solution for `largest_numbered_line.py`

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

# print line of a file containing largest number
# written by d.brotherston@unsw.edu.au for COMP(2041|9044)

import sys
import re

lines = sys.stdin.readlines()
numbers = []
for line in lines:
    numbers.append(list(map(float, re.findall(r'[+-]?(?:[0-9]+\.?[0-9]*|\.[0-9]+)', line))))

M = []
for line in numbers:
    try:
        M.append(max(line))
    except ValueError:
        pass

try:
    M = max(M)
    for index, line in enumerate(numbers):
        if M in line:
            print(lines[index], end='')
except ValueError:
    pass
```

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 8 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.

COMP(2041|9044) 23T2: Software Construction is brought to you by
the School of Computer Science and Engineering
at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs2041@cse.unsw.edu.au
CRICOS Provider 00098G