

COMPUTER SCIENCE 501/590C
Computing for Science and Engineering
FALL 2008
ASSIGNMENT # 1 (32 points)
September 1

Due Saturday, September 13 at 6:00 am Eastern Time This assignment covers through Section 3.1 of the class notes. The first question exercises logical thinking skills.

1. (3 points) Let x and y be “true values” and let δx and δy be relative errors introduced into x and y , respectively.
 - (a) Obtain an expression for the value that approximates x . Similarly for y . Your expressions *must* be in terms of the four variables defined above.
 - (b) Let $z = f(x, y)$. Give an expression for the relative error δz in an approximation to z obtained from approximate values of x and y . Your expression *must not* use any variable symbols other than x , y , δx , δy , and f .
 - (c) What equation in the notes can be used to check your answer to part (b)? And how would you perform this check?
2. (1 point) What is the effect of `chmod 644 myfile?` of `chmod 600 myfile?`

3. (5 points) Write a python script `interest` which takes three arguments: (i) current balance in dollars, (ii) annual rate of interest in percent, and (iii) number of days. It prints on one line the amount of interest and on another line the new balance. Here is an example:

```
$ ./interest 2000.34 11.99 22
interest = 13.66
new balance = 2014.00
```

Assume a leap year. The daily rate of interest is then $(1 + APR/100)^{1/366} - 1$, which is to be compounded daily. You may use floating-point arithmetic, but print the results to two decimal places.

4. (14 points) Create a module `numerals.py` containing functions `rom2arab(rom)`, which converts a roman numerals string to arabic, and `arab2rom(arab)`, which does the opposite. Roman numerals use the seven letters I, V, . . . , M. Both input and output are to be **strings**. Assume the use of new-style Roman numerals, which avoids four consecutive identical symbols—except that there is no limit to the number of repeated M’s. However, as an intermediate step in the conversion, it is recommended that you use old-style Roman numerals, and convert between them and new-style numerals using the function `str.replace`, e.g.,

```
rom = str.replace(rom, 'VIII', 'IX')
```

There are altogether six such replacements and the ordering of them matters! For one of `rom2arab` and `arab2rom`, it is convenient to use a **dictionary**, but for the other, it is easier to use two lists. (The reason has to do with the order in which a `for` loop accesses the entries of a **dictionary**.) For looping through pairs of lists, the `zip` function is handy. For full credit, create functions for `numerals.py` that are reasonably concise. (It is fine if the conversion between old- and new-style roman is done case by case without a loop, but the conversion between old-style roman and arabic may lead to bloated code if it is done without enough looping.)

5. (1 point) Here is a step-by-step example of conversion:

```
1492
 492 M
 392 MC
 292 MCC
 192 MCCC
  92 MCCCC
  42 MCCCCCL
  32 MCCCCCLX
  22 MCCCCCLXX
  12 MCCCCCLXXX
   2 MCCCCCLXXXX
   1 MCCCCCLXXXXI
   0 MCCCCCLXXXXII
```

What is the loop invariant for the loop that does this conversion from arabic to old-style roman?

6. (8 point) Write a Python script `find.py` which takes two arguments (i) a valid path name and (ii) a string. It prints out all paths containing the string in the name of one of the directories/files in the path, one path per line. If the given path is relative, it prints relative paths. If the given path is absolute, it prints absolute paths. Here is an example:

```
$ find.py junk python
junk/Mypython
junk/ipython.txt
junk/Myperl/pl2python
junk/Mypython/current/python2.5
```

Your script must not use Unix commands but rather generic Python commands from the `os` module. In particular, see the example for `os.walk`. Also useful is `os.path.join` and `str.find`.

Send your solutions electronically as described below. You should send one PDF file containing your answers to problems 1, 2, and 5 as well as your source code for problems 3, 4, and 6. Additionally, send your source code as separate text files. PDF files can be created by scanning hard copy or by printing-to-PDF from a word processor or text editor or by saving-as or by using `pdflatex`. You may use plain text to represent mathematics, in which case use of simplified \LaTeX notation is recommended, e.g., caret for superscript and underscore for subscript.

1. Log on to Blackboard at <http://www.itap.purdue.edu/tlt/blackboard/> (To use Blackboard, you have to have Java enabled for your web browser.)
2. Click on Assignments from the left panel. You will find the homeworks.
3. Go into a homework assignment, you will see a submission window on the page. Below the window there is something called “select zip file”. Click the “Add Attachments” tab below it. You will be able to submit the `.zip` file containing your homework answers there.
4. Now due to the limitation of Blackboard, it accepts only `.zip` files and not other formats such as `.tar`, `.rar` or `.bz`. Therefore, you have to compress your homework answer to a `.zip` file first. There are two ways you can do it:

- (a) Zip on your own first and when it comes to adding the attachments in Blackboard, choose “My computer” and upload your zip file directly from the computer. (Note again, it has to be a .zip file, otherwise even if you can upload it, you’ll not be able to submit it. That’s the requirement of Blackboard.) To pack a set of files and directories into a single file, use

```
zip <name>.zip <list>
```

and to unpack

```
unzip <name>.zip
```

- (b) Go to “My Tools” on the bottom of the left panel. Click on “My Files”, and then you will see a page where you can manage all your files for this course, and zip your file using Blackboard. If you keep all your files for this course in Blackboard, when adding the attachment for the homeworks, choose from “My Files”.

5. Submission will be closed after the due time. Please be careful with your time management.

Partial testers

1. A tester for interest:

```
#!/usr/bin/env python
import os
ofile = os.popen('interest 2000.34 11.99 22')
print file.readline(ofile) == 'interest = 13.66\n'
print file.readline(ofile) == 'new balance = 2014.00\n'
```

2. A template with a tester for numerals.py

```
#!/usr/bin/env python
def rom2arab(rom):
    ...
def arab2rom(arab):
    ...
# assert ;
# what is the loop invariant of your program?
# give step by step example
if __name__ == '__main__':
    print arab2rom('1492') == 'MCDXCII'
    import random
    for i in range(8):
        arab = str(random.randrange(1, 5000))
        print rom2arab(arab2rom(arab)) == arab
```

3. A tester for find.py:

```
#!/usr/bin/env python
import os
if 'junk' not in os.listdir(os.getcwd()):
    os.makedirs('junk/Myperl')
    os.makedirs('junk/Mypython/current')
    file.write(open('junk/pythagoras', 'w'), 'trash\n')
    file.write(open('junk/ipython.txt', 'w'), 'trash\n')
```

```
file.write(open('junk/Myperl/pl2python', 'w'), 'trash\n')
file.write(open('junk/Myperl/pyth2pl', 'w'), 'trash\n')
file.write(open('junk/Mypython/current/python2.5', 'w'), 'trash\n')
ofile = os.popen('find.py junk python')
print file.readlines(ofile) == ['junk/Mypython\n', 'junk/ipython.txt\n',
    'junk/Myperl/pl2python\n', 'junk/Mypython/current/python2.5\n']
ofile = os.popen('find.py junk/Myperl pl')
print file.readlines(ofile) == ['junk/Myperl/pl2python\n',
    'junk/Myperl/pyth2pl\n']
```