logo_CDM.png

CSC 241: introduction to computer science i

midterm exam

Please put your name below and hand in this sheet with any notes you use during the exam.

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The exam is out of 100 points and the points for each question are listed next to the problem. You should begin by downloading the file **csc241midterm.py** from the zip file on the D2L site under the Exams section in the Content area. Write all of your answers into the file and upload the file when you have completed the midterm exam.

You have until 3 pm for the midterm exam. All submissions must be complete by that time in order to be considered. No late submissions will be considered for any reason. Please do not submit until you have completed the exam, since **the dropbox for the midterm is set up to only allow you to submit ONCE**.

The exam is closed book. You may use five double-sided sheets (8 ½ x 11 inches) or ten single-sided sheets (8 ½ x 11 inches) of notes. The sheet(s) of notes (with your name at the top) must be handed in at the conclusion of the midterm exam. You may not use any other written materials. You are allowed to use the IDLE editor and the Python shell during the exam. You are not allowed to access the Internet or use any other electronic devices including cell phones. You may not communicate with anyone other than Instructor Hop during the exam. Any communication of any form with a person other than Instructor Hop during the midterm exam indicates that you are done, and you will be required to immediately submit your exam. Please also be careful to keep your eyes on your own monitor at all times. The lab machines are located close together, and we don’t want any misunderstandings to occur.

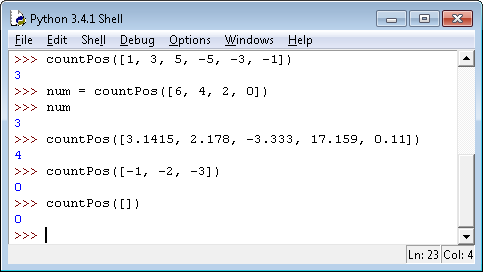
Please note that you may not find all problems equally easy. **If you get stuck on a problem, move on to the next one.**

Please also be aware that the sample executions and/or test cases are only examples.You may not make any assumptions about the information that the function will receive beyond what is specified in the problem description. **Your functions must work correctly on all valid inputs.**

#### **Question 1:** (25 points)

Write a function **countPos**() that takes a list of numbers as a parameter and **returns** the number of values in the list that are positive (> 0). If there are no positive numbers in the list, the function should return 0. You are allowed to assume that the function contains only numerical values.

The following shows the behavior of the function for some sample parameters:

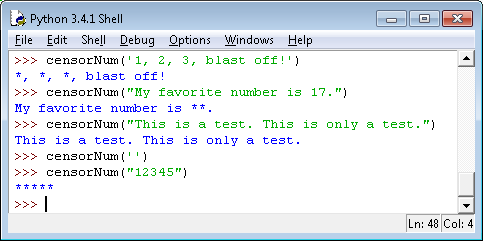


#### **Question 2:** (25 points)

Write a function **censorNum**() that takes a string as a parameter and **prints** a censored version of the string to the screen. The message printed to the screen has an asterisk in place of each digit appearing in the string. A digit is one of 1, 2, 3, 4, 5, 6, 7, 8, 9, or 0.

For example, if the function were called on the string '1, 2, 3, blast off!' then the message printed to the screen would be \*, \*, \*, blast off! If the function is called on a string with no digits, the sequence of characters in the original string should be printed. If the function is called on an empty string, nothing (not even a newline) should be printed.

The following shows several sample executions of the function:



#### **Question 3:** (25 points)

Write a function **apr**() that takes a string as a parameter. The string has the following format:

dept num grade

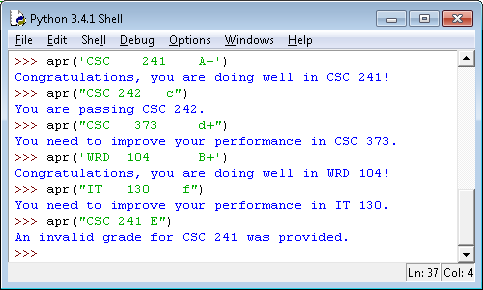
where dept is a three-character code for the department of the course (e.g. CSC), num is an integer with 2 or 3 digits representing the course number (e.g. 241), and grade is a letter grade, possibly including a plus or a minus (e.g. A-). Each of the three pieces of the string is separated by at least one (but possibly more than one) space.

The function **prints** a message to a student taking the course specified who is receiving the indicated grade. Plusses and minuses are not considered when determining what message to print. The message is based on the grade as follows, where the missing piece in the output (…) is the course department and number:

* If the student is earning an A or B, the function prints: Congratulations, you are doing well in …
* If the student is earning a C, the function prints: You are passing ….
* If the student is earning a D or F, the function prints: You need to improve your performance in …
* If another other grade is specified, the function prints: An invalid grade for … was provided.

Capitalization does not matter when specifying the grade. For example, a grade of A-, b+, or B should all result in the same greeting being printed. There should be at least one space printed between the department and the course number.

The following shows how the function would be used on several sample parameters:



#### **Question 4:** (25 points)

Implement a function **sentenceCount**() that takes a string representing the name of a file as a parameter and **returns** the number of sentences in the file. A sentence is a string of characters followed by a period (.), exclamation mark (!), or question mark (?).

To allow you to distinguish periods that end sentences and periods that end abbreviated words (like Mr. or Ph.D.) you can assume that every sentence ending with a period will have a newline immediately following the period. You should not make the same assumption about sentences ending with exclamation marks or question marks.

The following shows the execution of the function on several sample files that are included in the zip file containing the exam template:

