COSC 101, Practice Exam #2

Name: ____

Instructions and advice:

- Do not open the exam until instructed to do so.
- Write your name and circle your section time.
- You have 50 minutes to complete this exam; use your time wisely.
- There are 6 questions and a total of 0 points available for this exam. Don't spend too much time on any one question.
- If you want partial credit, show as much of your work and thought process as possible.
- Since indentation is important in Python, please be sure that your use of indentation is obvious for any code you write.
- When defining functions, it is not necessary to write docstrings nor is it necessary to write comments.
- If you run out of space while answering a question, you can continue your answer on one of the scrap pages at the end of the exam. If you do so, be sure to indicate this in two places: (1) below the question, indicate which scrap page contains your answer, and (2) on the scrap page, indicate which question you are answering.

Question	Points	Score
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
Total:	0	

1. Assume that the following statements have already been executed:

```
a = 'hamilton'
b = 2
c = """New York"""
d = -3
```

For each of the following expressions, evaluate the expression and write the resulting value, or identify the error in the code that would prevent it from running.

(a) `c[1]`

Solution:

```
'e'
```

(b) `c[d]`

Solution:

```
'o'
```

(c) `a[:b]`

Solution:

```
'ha'
```

(d) `c[5] in a`

Solution:

```
True
```

(e) `a[-1] not in c`

Solution:

```
True
```

(f) `a[len(c)]`

Solution:

```
Error: string index out of range
```

(g) `c.lower()`

Solution:

```
'new york'
```

(h) `c[b:d]`

Solution:

```
'w Y'
```

2. (a) What is the output of the following program?

```
def grow_leaves(x, y):
    z = x + ' has ' + y
    return z
    print('nuts!')

x = 'maple'
y = 'leaves'
z = 'squirrel!'
print(x + ' has ' + y)
grow_leaves(y, z)
print(z)
print(x + ' has ' + y)
```

Solution:

```
maple has leaves
squirrel!
maple has leaves
```

(b) What is the output of the following program?

```
for i in range (3):
    print ("->", end='')
    for j in range (i+1):
        print (i, end='')
    print ("<-")
```

Solution:

```
->0<-
->11<-
->222<-
```

(c) What is the output of the following program?

```
for i in range(2):
    for j in range(3):
        print(i, end='')
    print()
    for j in range(3):
        print(i+1, end='')
    print()
```

Solution:

```
000
111
111
222
```

(d) What is the output of the following program?

```
s = "code"
counter = 0
for char in s:
   if counter % 3 == 0:
       print('!')
   counter = counter + 1
   print(char, end='')
   if counter % 2 == 0:
       print('*')
```

Solution:

```
!
co*
d!
e*
```

3. Suppose we want to write a program that does DNA analysis. While real DNA is made up of the nucleotides guanine, adenine, thymine, and cytosine, for our purposes, we can think of DNA as simply a string that contains only the letters G, A, T, and C.

   Write a program that asks a user for a snippet of their DNA and checks whether it is a *valid* DNA sequence. A sequence is valid if the only letters that occur are the letters G, A, T, and C. These letters can appear in any order and any number of times (including not at all). Your program should print `"Yes"` if it is valid and `"No"` if it is not. For example, if the user types `"GATCGGAXC"` then your program should print `"No"` because the `"X"` is not a valid DNA letter. On the other hand, if the user types `"GGCTCT"`, your program should print `"Yes"`.

Solution: Solution that uses a boolean flag and a search pattern:

```
dna = input("Enter a DNA string: ")
validdna = True
for char in dna:
   if char != 'A' and char != 'C' and char != 'G' and char != 'T':
       validdna = False

if validdna:
   print("Yes")
else:
   print("No")
```

Alternative solution that uses accumulator pattern:

```
dna = input("Enter a DNA string: ")
count_valid = 0
for char in dna:
   if char == 'A' or char == 'C' or char == 'G' or char == 'T':
       count_valid = count_valid + 1

if count_valid == len(dna):
   print("Yes")
else:
   print("No")
```

4. Write a program that asks for a string from the user, as well as a single character (a one-character string). Find and print the index of the first occurrence of the character within the string. If the character does not appear in the string, print -1.

   For example, if a user types "invalid" and "i", your program should print 0 (zero), since that is the first index where the character "i" occurs. As another example, if a user types "555-1212" and "0", your program should print -1 since the character "0" is not present in "555-1212".

Solution:

```
s = input("Enter a string: ")
char = input("Enter one character: ")
index = -1
for i in range(len(s)):
        if s[i] == char and index == -1:
                index = i
print(index)
```

5. A young entrepreneur starts a job that, although it pays her only 1 dollar on the first day, her salary doubles every day. So on her second day of work, she is paid 2 dollars; on her third day of work, she is paid 4 dollars, etc. This entrepreneur also generously gives 25% of her daily salary to charity. Thus, on day 1, her take-home pay is $0.75; on day 2, her take-home pay is $1.50; on day 3, her take home pay is $3, etc.

   How many days must she work before her take-home daily pay (daily salary minus charity) reaches a certain goal? Write a function `countDays` that takes as input an integer parameter `goal` and returns the fewest number of days she must work to have a daily take-home pay of at least `goal` dollars.

   Examples:

   - `countDays(0.50)` should return 1 because on day 1 she takes home $0.75.
   - `countDays(1.51)` should return 3 because she only takes home $1.50 on the second day, so must work a third day to reach the goal of $1.51.
   - `countDays(3)` should return 3 because after her third day of work her daily salary is $4 and she takes home exactly $3 dollars.

Solution:

```
def countDays(goal):
    """
    >>> countDays(.75)
    1
    >>> countDays(3)
    3
    """
    salary = 1
    days = 1
    while salary * .75 < goal:
        salary *= 2
        days += 1
    return days
```

6. Write a function called `capThese(s, char)` that takes two parameters, a string `s` composed of at least 1 character, and a string `char` composed of exactly 1 letter, which may be lower or upper case. The function should return a new string in which all the occurrences of the letter `char` in `s` are capitalized.

   For example, `capThese("moo cow, moo", "o")` should return the string `"mOO cOw, mOO"`, and `capThese("me love cookies!", "E"` should return the string `"mE lovE cookiEs!"`.

   For this problem, the only built-in string methods you may use are `upper` and `lower`. Recall also that `ord(c)` returns the numeric equivalent of a character `c`, while `chr(x)` returns the character equivalent of the number `x`.

Solution:

```python
def capThese(s, char):
    newstr = ""
    char = char.lower()
    for letter in s:
        if letter == char:
            newstr += letter.upper()
        else:
            newstr += letter
    return newstr
```

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)