CSCI 121: Computer Science Fundamentals I
(Practice) Midterm Exam – Spring 2025

This is a practice exam for the in-class midterm that will be held on Monday, March 17th.

----

1. For each of the following Python scripts, write what the script would output to the console. If some script line causes an error, write *Error!* and stop running the script, similar to what Python would do.

   (a)
   ```
   a = 7
   b = 3
   c = "4"
   print(a // b)
   print(a % b)
   print(b // a)
   print(b % a)
   print(c * b)
   print(c + b)
   ```

   (b)
   ```
   i = 10
   j = 7
   while i > 5:
       while i > j
           j = j + 1
           i = i - 2
           print(str(i) + " " + str(j))
       j = j - 5
   ```

   (c)
   ```
   def g(x):
       print(x)
       return x*x

   def f(a,b):
       return g(a) + g(b)

   print(g(f(1,2)))
   ```

   (d)
   ```
   def h(xs,a):
       xs = xs + [a]
       print(xs)

   bs = [1,2,3]
   print(bs)
   h(bs,4)
   print(bs)
   h(bs,5)
   print(bs)
   ```

2. Beatrice drives a taxi. She charges riders a base cost of `$3` and then an additional `$2` per mile driven if the ride is 10 miles long or shorter. If the ride is more than 10 miles, there is no base cost and the charge is `$4` per mile. Write a function `taxi_cost` that takes an integer parameter giving the length of a trip in miles. It should return Beatrice's taxi charge for that trip. For example:

```
>>> taxi_cost(2)
7
>>> taxi_cost(6)
15
>>> taxi_cost(10)
23
>>> taxi_cost(12)
48
>>> taxi_cost(15)
60
```

3. Every positive integer can be written as a product of a power of two times an odd integer. For example:

$$\begin{aligned} 24 &= 8 \cdot 3 = 2^3 \cdot 3 \\ 100 &= 4 \cdot 5 = 2^2 \cdot 5 \\ 7 &= 1 \cdot 7 = 2^0 \cdot 7 \end{aligned}$$

This means that 3 is the odd factor of 24, 5 is the odd factor of 100, and 7 is the odd factor of itself. Write a function `odd_factor` that takes a positive integer and returns its odd factor.

```
>>> odd_factor(24)
3
>>> odd_factor(100)
5
>>> odd_factor(7)
7
```

4. Write a procedure `reverse` that takes a list and reverses it.

```
>>> xs = [1, 2, 3, 4]
>>> reverse(xs)
>>> xs
[4, 3, 2, 1]
```

5. Write a procedure `describe` whose first parameter is a dictionary whose keys are names (as strings) and whose associated values are ages (positive integers). Its second parameter is an integer number of columns.

   It should output each of the dictionary's entries, line by line, formatting it so that each line has the same number of columns of text, with each name and age separated by periods, as shown below:

   ```
   >>> d = {'Jim': 54, 'Bea': 2, 'Berni': 83}
   >>> describe(d,10)
   Jim.....54
   Bea......2
   Berni...83
   ```

   You can assume that the number of columns requested will lead to an output of at least one period on each line.