CSc 110 Sample Midterm Exam #1

1. Expressions

For each expression in the left-hand column, indicate its value in the right-hand column.
Be sure to list a constant of appropriate type (e.g., `7.0` rather than `7` for a `float`, `String`s in quotes).

Expression	Value
`8 + 5 * 3 / 2`	_____
`1.5 * 4 * 7 // 8 + 3.4`	_____
`73 % 10 - 6 % 10 + 28 % 3`	_____
`4 + 1 + 9 + (-3 + 10) + 11 // 3`	_____
`3 // 14 // 7 / (1.0 * 2) + 10 // 6`	_____
`10 > 11 == 4 / 3 > 1`	_____
`not(2 >= 11 or 10 < 67 or 4 / 4 >= 1)`	_____
`(True or not 2 < 3) and 6 == 4 / 3`	_____

2. Parameter Mystery

At the bottom of the page, write the output produced by the following program.

```
def main():
    x = "happy"
    y = "pumpkin"
    z = "orange"
    pumpkin = "sleepy"
    orange = "vampire"

    orange(y, x, z)
    orange(x, z, y)
    orange(pumpkin, z, "y")
    z = "green"
    orange("x", "pumpkin", z)
    orange(y, z, orange)

def orange(z, y, x):
    print(y, "and", z, "were", x)
```

3. If/Else Simulation

For each call of the function below, write the value that is returned:

```
def mystery(n):
    if n < 0:
        n = n * 3
        return n
    else:
        n = n + 3
    if n % 2 == 1:
        n = n + n % 10
    return n
```

Function Call	Value Returned
`mystery(-5)`	_____
`mystery(0)`	_____
`mystery(7)`	_____
`mystery(18)`	_____
`mystery(49)`	_____

4. Programming

Write a function named `month_apart` that accepts four integer parameters representing two calendar dates. Each date consists of a month (1 through 12) and a day (1 through the number of days in that month [28-31]). Assume that all dates occur during the same year. The method returns whether the dates are at least a month apart. For example, the following dates are all considered to be at least a month apart from 9/19 (September 19): 2/14, 7/25, 8/2, 8/19, 10/19, 10/20, and 11/5. The following dates are NOT at least a month apart from 9/19: 9/20, 9/28, 10/1, 10/15, and 10/18. Note that the first date could come before or after (or be the same as) the second date. Assume that all parameter values passed are valid.

Sample calls:

```
month_apart( 6, 14,  9, 21) should return True,  because June 14 is at least a month before September 21
month_apart( 4,  5,  5, 15) should return True,  because April 5 is at least a month before May 15
month_apart( 4, 15,  5, 15) should return True,  because April 15 is at least a month before May 15
month_apart( 4, 16,  5, 15) should return False, because April 16 isn't at least a month apart from May 15
month_apart( 6, 14,  6,  8) should return False, because June 14 isn't at least a month apart from June 8
month_apart( 7,  7,  6,  8) should return False, because July 7 isn't at least a month apart from June 8
month_apart( 7,  8,  6,  8) should return True,  because July 8 is at least a month after June 8
month_apart(10, 14,  7, 15) should return True,  because October 14 is at least a month after July 15
```

5. Programming

Write a function named `print_grid` that accepts two integer parameters *rows* and *cols*. The output is a grid of numbers where the first parameter (*rows*) represents the number of rows of the grid and the second parameter (*cols*) represents the number of columns. The numbers count up from 1 to (*rows* x *cols*). The output are displayed in column-major order, meaning that the numbers shown increase sequentially down each column and wrap to the top of the next column to the right once the bottom of the current column is reached.

Assume that *rows* and *cols* are greater than 0. Here are some example calls to your function and their expected results:

Call	`print_grid(3, 6)`	`print_grid(5, 3)`	`print_grid(4, 1)`	`print_grid(1, 3)`

Output	`1 4 7 10 13 16`
	`2 5 8 11 14 17`
	`3 6 9 12 15 18`
		`1 6 11`
		`2 7 12`
		`3 8 13`
		`4 9 14`
		`5 10 15`
			`1`
			`2`
			`3`
			`4`
				`1 2 3`

6. Programming

Write a function named `count_even_digits` that accepts two integers as parameters and returns the number of even-valued digits in the first number. An even-valued digit is either 0, 2, 4, 6, or 8. The second value represents how many digits the number has. The second value is guaranteed to match the number of digits in the first number.

For example, the number 8546587 has four even digits (the two 8s, the 4, and the 6), so the call `count_even_digits(8346387, 7)` should return `4`.

You may assume that the values passed to your function are non-negative.

CSc 110 Sample Midterm Exam #1 Key

1. Expressions

Expression	Value
`8 + 5 * 3 / 2`	`15.5`
`1.5 * 4 * 7 // 8 + 3.4`	`8.4`
`73 % 10 - 6 % 10 + 28 % 3`	`-2`
`4 + 1 + 9 + (-3 + 10) + 11 // 3`	`24`
`3 // 14 // 7 / (1.0 * 2) + 10 // 6`	`1.0`
`10 > 11 == 4 / 3 > 1`	`False`
`not (2 >= 11 or 10 < 67 or 4 / 4 >= 1)`	`False`
`(True or not 2 < 3) and  6 == 4 / 3`	`False`

2. Parameter Mystery

```
happy and pumpkin were orange
orange and happy were pumpkin
orange and sleepy were y
pumpkin and x were green
green and pumpkin were vampire
```

3. If/Else Simulation

Function Call	Value Returned
`mystery(-5)`	`-15`
`mystery(0)`	`6`
`mystery(7)`	`10`
`mystery(18)`	`22`
`mystery(49)`	`52`

4. Programming (four solutions shown)

```
def month_apart(m1, d1, m2, d2):
    if m1 == m2:
        return False
    elif m1 <= m2 - 2:
        return True
    elif m1 >= m2 + 2:
        return True
    elif m1 == m2 - 1:
        if d1 <= d2:
            return True
        else:
            return False
    elif m1 == m2 + 1:
        if d1 >= d2:
            return True
        else:
            return False
    else:
        return False
```

```
def month_apart(m1, d1, m2, d2):
    if m1 < m2 - 1 or m1 > m2 + 1:
        return True
    elif m1 == m2 - 1 and d1 <= d2:
        return True
    elif m1 == m2 + 1 and d1 >= d2:
        return True
    else:
        return False


def month_apart(m1, d1, m2, d2):
    return (m2 - m1 > 1) or (m1 - m2 > 1) or
           (m2 - m1 == 1 and d1 <= d2) or
           (m1 - m2 == 1 and d1 >= d2)


def month_apart(m1, d1, m2, d2):
    return abs((m1 * 31 + d1) - (m2 * 31 + d2)) >= 31
```

5. Programming (one solutions shown)

```
def print_grid(rows, cols):
    for i in range(1, rows + 1):
        for j in range(cols):
            print(i + rows * j, "", end='')
        print()
```

6. Programming

```
def count_even_digits(n, length):
    count = 0
    for i in range(0,length):
        digit = n % 10
        n = n // 10
        if (digit % 2 == 0):
            count += 1

    return count
```