Spring 2020 COP 2930 Final Exam Part A (April 27, 2020) Solution

1) (5 pts) The following line of python code has a syntax error. How can it be fixed?

```
value = int(input("How much is the item worth?\n")
```

Solution
Place an additional closing parenthesis at the end for a total of two closes to match the two opens:

```
value = int(input("How much is the item worth?\n"))
```

Grading: 5 pts, mostly all or nothing

2) (6 pts) Python assigns types to variables based on how they are first used. What are the types assigned to the variables a, b, and c in the lines of code below?

```
a = input("")
b = int(input(""))
c = float(input(""))
```

Solution
a is of type String
b is of type int
c is of type float

Grading: 2 pts for each

3) (8 pts) A programmer is trying to read in a menu choice which must be an integer in between 1 and 5. If it is not, she wants to reprompt the user to enter a valid choice. What is wrong with her code segment below? How can it be fixed? (Don't answer the brevity of the print message. I am only concerned with what the code does, not how pretty or user friendly it is.)

```
choice = 0
while choice < 1 and choice > 5:
    choice = int(input("Please enter a valid choice."))
```

Solution
The loop never enters because it's impossible for any variable to both be less than 1 and greater than 5 at the same time. It can be fixed by changing the and to an or.

Grading: Give full credit if they say to change and to or. If they say the loop is never entered but don't suggest the fix, give 5 of 8 points.

4) (9 pts) Provide three examples of mod (%) providing a useful calculation.

Solution
1) Converting from inches to feet and inches.
2) Keeping track of whose turn it is in a traditional game where you start with player 0, then player 1, and after player n-1 goes, player 0 goes again.
3) Determining if one integer is divisible by another.

Grading: 3 pts for each use, many possible answers

5) (10 pts) A student is trying to figure out how many 3 ft by 3 ft tiles can completely fit (without any cutting of tiles) on a floor of size width feet by length feet and has written the code shown below. It is incorrect. How can he fix it?

```
width = int(input("What is the width of the room\n"))
length = int(input("What is the length of the room in feet?\n"))
tiles = int((width/3)*(length/3))
print(tiles, "complete tiles can fit on the floor.")
```

Solution
The code can be fixed by changing both of the division operations to integer division. Specifically, this is how the third line of code should be:

```
tiles = (width//3)*(length//3))
```

Notice that once we do integer division, the conversion from float to int is no longer necessary (but can be kept if one wants to).

Grading: Give 5 pts if they say to divide area by 9, give 8 pts if they say to integer divide area by 9 (this is actually incorrect but closer), give full credit for the correct response. Grader's discretion for all answers.

6) (12 pts) The following function takes in a list of lists, where each list represents a list of one student's grades. The job of the function is to replace the minimum grade in the list with a -1 as a way of indicating that that grade should be skipped. The function does not work. Why? What can be done to fix it? (Hint: The fix actually shortens the code a bit!)

```
def dropMinGrade(studentGrades):

    for i in range(len(studentGrades)):

        minIdx = 0
        for j in range(len(studentGrades[i])-1):
            if studentGrades[i][j] > studentGrades[i][j+1]:
                minIdx = j
            else:
                minIdx = j+1

        studentGrades[i][minIdx] = -1
```

Solution

The function doesn't work for two reasons:

1) It tries to identify the index storing the maximum value in the list instead of the minimum.
2) It incorrectly tries to identifies that index.

Typically, when looking for a min or max, there should be an if statement without an else inside of a loop, since no action should be taken if the current value doesn't exceed the running min or max. So the else is not good since minIdx should NOT always be updated.

Secondly, we want to update minIdx if the current value is LOWER than the running min, not higher than it, as is done in the code above. Here is one possible fix to the portion of the code inside the i loop:

```
        minIdx = 0
        for j in range(len(studentGrades[i])):
            if studentGrades[i][j] < studentGrades[i][minIdx]:
                minIdx = j
```

So, we must (1) change the sign of the inequality, (2) loop through the whole array, or loop from index 1 to index len(studentGrades[i])-1, inclusive, (3) replace j+1 with minIdx, since we want to compare our current value with the minimum so far, not the next value.

Grading: 4 pts for min/max issue, 4 pts for if/else, 4 pts for appropriate changes