Fall 2020 COP 2930 Final Exam Part A

Date: 12/8/2020
Time Given: 10:00 am
Time Due: 10:50 am
Late Due Time: 11:00 am

Directions: Please use your course notes and a calculator as aids for this exam. Do NOT attempt to look up information online outside of the course webpage and do NOT run things in IDLE. Please type your answers directly in the provided .docx file, or type your answers in your own text file or .docx file. The accepted file types for submission will be .doc, .docx, .txt and .pdf.

It's recommended that you stop working at the due time and start uploading at that time. Anything turned in before the late due time will be accepted for full credit. Anything that doesn't make it in by the late due time will earn a 0. A 10 minute buffer is provided after both due times. Please don't take advantage of these buffers as it's an unnecessary risk.

1) (6 pts) A juice costs 75 cents and a soda cost 90 cents. Complete the program below so that it asks the user how many juices to buy and how many sodas to buy and prints out the number of dollars and cents (in between 0 and 99, inclusive) those juices and sodas will cost. (There is no tax.) Note that the way the given code is, you must use variables named dollars and cents in your solution. (You may use other variables as well, if you would like to.)

```
numJuice = int(input("How many juices to buy?\n"))
numSoda =  int(input("How many sodas to buy?\n"))

# Put your code here.

print("Total cost =",dollars,"dollars and",cents,"cents.")
```

2) (8 pts) In other programming languages, there is a "matching-else" problem, where if there are two nested if's but only one corresponding else, there is some perceived ambiguity as to which if the else matches to. In Python, this problem does not exist. What rule is used in Python to determine which else a particular if matches to? Show an example where there are two nested if's and one else, explain what that example does and (a) how one can tell which if the one else matches and (b) why you designed the else to match there.

3) (8 pts) A simple way to find the largest integer less than or equal to the cube root of a given positive integer, *n*, is to start with the integer 1, cube it, and compare it to *n*. If this cube is less than or equal to *n*, then we add one to our integer and repeat the process. If this cube is greater than *n*, we stop our loop and know the answer is one less than the number we last tried. Here is a quick example of the algorithm running for $n = 400$

$1^3 = 1 \leq 400$, so go to the next integer
$2^3 = 8 \leq 400$, so go to the next integer
$3^3 = 27 \leq 400$, so go to the next integer
$4^3 = 64 \leq 400$, so go to the next integer
$5^3 = 125 \leq 400$, so go to the next integer
$6^3 = 216 \leq 400$, so go to the next integer
$7^3 = 343 \leq 400$, so go to the next integer
$8^3 = 512 > 400$, so we STOP!
Since 8 was the first number that was too big, our correct answer is 7.

Complete the code below so that it simulates this process and finds the largest integer less than or equal to the cube root of n (entered by the user) and prints this value out to the screen. You must use a variable res as shown below, but you may use any additional variables, if you like.

```
n = int(input("Please enter a positive integer.\n"))

# Put your code here.

print("Largest integer <= cuberoot of",n,"is",res)
```

4) (8 pts) Complete the segment of code below so that it prints out a Manhattan Distance Grid, showing the Manhattan Distances from the top left corner of the grid. Let the user enter the number of rows and columns for the grid. The Manhattan distance is defined as the sum of the zero based row and column number of a location. Here is the grid that would print out for 3 rows and 4 columns:

```
0    1    2    3
1    2    3    4
2    3    4    5
```

Separate out each item in a row with a tab. Please fill in the code segment started below:

```
numRows = int(input("How many rows for your Manhattan Grid?\n"))
numCols = int(input("How many columns for your Manhattan Grid?\n"))

# Put your code here.
```