COSC 1336 Lab: 4

Relevant reading: Chapter 4

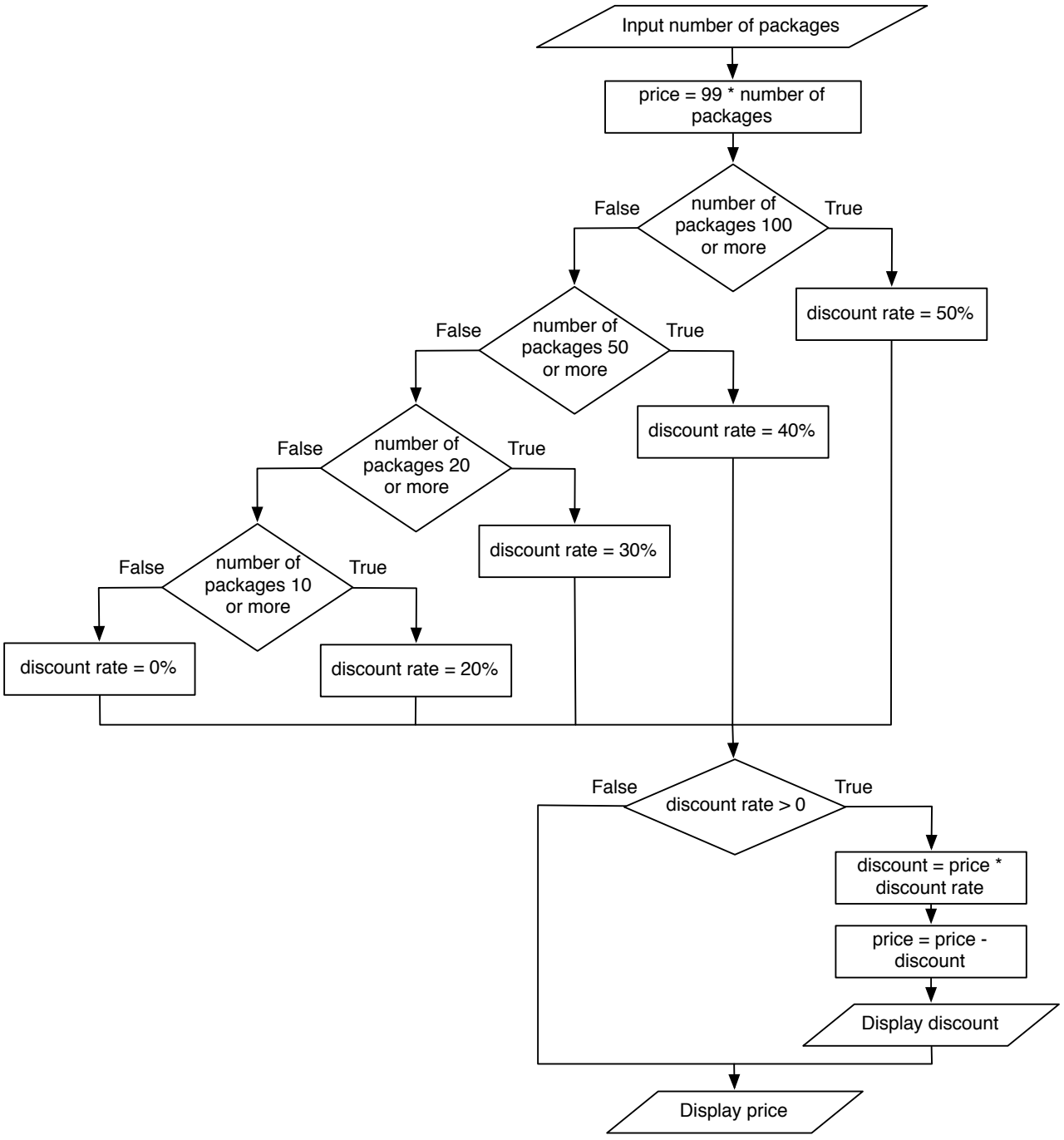
Due: Oct. 2, 2:30 pm

(Late date: Oct. 9, 2:30 pm)

50 Points

Problem 1. [10 points] Software Sales: Read Programming Exercise #8 on p. 154 of the textbook. For this problem you will create two different Python implementations of the described program.

a. **[5 points]** The flowchart below shows a design for the program that uses nested `if` and `else` structures. In a file called `software_sales_nested.py`, create a Python implementation of the design shown in the flowchart. Be sure that your implementation is an accurate reflection of the design, and write the program in a `main` function.



b. **[5 points]** In a file called `software_sales_elif.py`, make another implementation of the same program that uses a `if/elif/else` structure. The behavior of the program should be exactly the same as the one from part a, but the program implementation should contain no nested `if` structures.

Problem 2. [12 points] Shipping Charges: Read Problem Exercise #9 in the textbook. Download the file `shipping_charges_buggy.py` from Blackboard. In it you will find a buggy version of the program described in the problem. Fix the bug in two different ways:

a. **[2 points]** First, in a comment toward the top of the file, give an example test case that causes the program to produce incorrect output. Say what the correct output should have been.

Now make two copies of the buggy `print_total_due` function, and call one of them `print_total_due_logical` and the other `print_total_due_elif`.

b. **[5 points]** In the function `print_total_due_logical`, fix the bug by changing the conditions of each `if` statement to use logical operators to fully specify the conditions in which the block of code should be executed. You should not need to change the selection structure being used, just the conditions. Make sure you change the function call in `main` to test your bug fix.

c. **[5 points]** In the function `print_total_due_elif` fix the bug by using an `if/elif/else` structure in place of the several independent `if` structures that are used currently. You should not need to change the conditions for this fix, just the selection structure. Again, test your bug fix by calling the `print_total_due_elif` function from `main`.

Problem 3. [20 points] Color Mixer: Read Programming Exercise #5 in the textbook. For this problem you will create two different Python implementations of the described program.

a. **[10 points]** First, create a program that uses a nested `if` structure. Other than the requirement that your program behave correctly and contain at least one nested `if`, you are free to design this however is most intuitive to you. I recommend doing this problem before reading part b so that you are not unduly swayed by the implementation described there.

b. **[10 points]** Now create a program that uses an `if/elif/else` structure, and contains only one block of code for each output the program will need to generate. To do this, you will need to use compound Boolean expressions as the conditions for each block of code. The following pseudocode describes the overall structure:

```
if at least one color is invalid:
   display an error message
else if the colors are the same:
   display the color
else if ????:
   display purple
else if ????:
   display orange
else:
   display green
```

Problem 4. [8 points] From Blackboard, download the `bmi_calc_improved.py` file. Right now it is the BMI calculator program that we wrote in lecture. Modify the program so that instead of printing out the whole table of BMI values and their interpretations, the program displays the appropriate interpretation for the user's BMI. You should only need to modify the `main` and `calc_display_bmi` functions (it's okay to have functions defined that are not called anywhere).