ENGR 102 – Summer 2022
Lab: Topic 4 (team) 26 points = 100 %

Deliverables.
This lab consists of three team activities. Please submit the following files to zyBooks.

- `make_change.py` [8 points]
- `pretty_equation.py` [10 points]
- `boolean_expressions.py` [8 points]

Please include the team header at the top of **each** file.
But this time, each of your team members will need to submit it on ZyBook.

Activity #1: Make change – team
Write a program that computes the change made when someone buys something. Your program should prompt the user to input from the keyboard how much they paid and how much it costs. Your program should then output the change and list how many of each type of coin. You may assume the change is always less than $1.

Although this may seem like an easy task, it can be quite challenging to get right. You MUST *think before you code*. Discuss with your team how to approach the problem and develop an algorithm (a sequence of steps to perform a computation or solve a problem) for any possible input. This task may take up to 30 minutes. It will take much longer if you do not think before you code.

First, your team should develop and document an algorithm for solving this problem. You may choose to write it by hand, type it in a document, draw a picture, etc, but you do not have to turn it in. Manually check your algorithm and check your answers versus the examples given below as well as others you come up with. You should not code until you develop and check your algorithm by hand. **After you develop an algorithm**, translate it into Python code and name your file make_change.py. Verify that your code delivers the correct output for all of the examples you thought up.

Your program should perform the following general tasks:

- Store input in appropriately named variables
- Convert the input to a number if you plan to perform mathematical operations with the input value
- Perform necessary decisions/procedures/calculations and store results in appropriately named variables
- Output results to the screen using the format below

Example output (using inputs **1**, **0.65**):

```
How much did you pay? 1
How much did it cost? 0.65
You received $0.35 in change. That is...
1 quarter
1 dime
```

Example output (using inputs **2**, **1.02**):

```
How much did you pay? 2
How much did it cost? 1.02
You received $0.98 in change. That is...
3 quarters
2 dimes
3 pennies
```

Hint: You may encounter a case where your code doesn't calculate the correct change. It may be helpful to go through Activity #2 (Roundoff error) in the individual lab.

Activity #2: Pretty equation – team

A quadratic equation is an equation of the form $Ax^2 + Bx + C = 0$ where A, B, and C are the coefficients of the equation. Write a program named pretty_equation.py that takes as input the coefficients A, B, and C and outputs the quadratic equation in a nice format. Your program should not print the coefficient "1" or a term with a coefficient "0" and should replace plus signs (+) with minus signs (-) for negative coefficients. Include one space around the sign and the term. You may want to review the string formatting document posted with Lecture 3. Use the output format shown below. You may assume the user only enters integers.

Example output (using inputs **-1**, **-2**, **3**):

```
Please enter the coefficient A: -1
Please enter the coefficient B: -2
Please enter the coefficient C: 3
The quadratic equation is - x^2 - 2x + 3 = 0
```

Activity #3: Boolean expressions – team

The purpose of this program is to give you practice with performing Boolean logic with Python. Create a file named boolean_expressions.py for this activity. Each team member is responsible for knowing this material, and it may be a good idea for all team members to practice keying in the code given. You are required to include code for Part A, Part B, and Part C below in your file for submission. Part D is optional and should be included in your file if you attempt it. *Please separate the various parts of your code with the following comment to identify the separate sections (copy/paste into your file with the appropriate letter)*.

```
############ Part A ############
```

Part A: Inputting Boolean values from the keyboard

Write a simple code to take as input from the user Boolean values from the keyboard for variables `a`, `b`, and `c`. The user should be able to enter a value indicating Boolean True and Boolean False from the keyboard and have the code create a Boolean variable representing the user's input for use in Boolean expressions that appear later in the code. You will need a simple code to make some decisions based on the values of these Boolean variables to verify that your scheme works.

The user should be able to enter `True`, `T`, or `t` for Boolean `True`. To indicate Boolean `False`, the user should be able to enter `False`, `F`, or `f`. Note: Numeric input is **not** permissible.

If this proves to be a bit of a challenge, review Lecture 3 slides covering Boolean conversions. You will need to use your new-found knowledge of `if-elif-else` blocks to make this work as described.

Part B: Evaluating Booleans

Add to your program code that evaluates the following Boolean expressions using the variables `a`, `b`, and `c` from Part A. The program should output the value `True` or `False` of the expression for the entered values. Use Boolean expressions; do **NOT** use `if-elif-else` blocks. See the example output below.

1. a and b and c
2. a or b or c

Part C: Writing Boolean expressions

Extend your program above to include Boolean expressions (do **NOT** use `if-elif-else` blocks) that meet the criteria in each item below using the variables `a`, `b`, and `c` from Part A. Your program should output the value `True` or `False` based on the previously entered values.

1. Given values for two Boolean variables a and b, create a Boolean expression for "exclusive or" or "XOR" between a and b. This expression evaluates to `True` if just one of a or b is True, but not if both are True or both are False. Do **NOT** use `^`, instead use **only** `not`, `and`, and `or` to construct an equivalent statement.
2. Given values for three Boolean variables a, b, and c, create a Boolean expression to determine if `True` was entered an odd number of times. This expression evaluates to `True` if exactly 1 or 3 of the variables a, b, and c is True, and is `False` otherwise. Use **only** `not`, `and`, and `or`.

Example output (using inputs **T**, **T**, **T**):

```
Enter True or False for a: T
Enter True or False for b: T
Enter True or False for c: T
a and b and c: True
a or b or c: True
XOR: False
Odd number: True
```

Part D: Other Activities (Optional 1 bonus point)

- Part D is strongly suggested for those wanting to major in Computer Science, Computer Engineering, Electrical Engineering, or another computationally heavy field.
- If you would like to learn more about Boolean Expression Simplification, follow this link to see the list of rules used for the Boolean expression simplifications http://sandbox.mc.edu/~bennet/cs110/boolalg/rules.html.
- Keep in mind that multiplication represents Boolean **and** and addition represents Boolean **or**. That means **AB** is equivalent to **A and B**, **A+B** is equivalent to **A or B**, and **A-bar** is equivalent to **not A**.
- Then take a look at the examples at http://sandbox.mc.edu/~bennet/cs110/boolalg/simple.html. Note that T stands for True.

Add to your program above using the variables `a`, `b`, and `c` from Part A and evaluate the following Boolean expressions.

1. (not (a and not b) or (not c and b)) and (not b) or (not a and b and not c) or (a and not b)
2. (not ((b or not c) and (not a or not c))) or (not (c or not (b and c))) or (a and not c) and (not a or (a and b and c) or (a and ((b and not c) or (not b))))

Then, create simpler (reduced) versions that still give the same results. Your program should output the value `True` or `False` based on the previously entered values. Output the results of the complex expression and the simple version. If your team completes Part D correctly, you will receive 1 bonus point on this assignment.

Part D example output (using inputs **T**, **T**, **T**):

```
Complex 1: False
Complex 2: True
Simple 1: False
Simple 2: True
```

Hint: There are 2 possible values for each of a, b, and c. Any expression that evaluates the same for all possible combinations of `True`/`False` of those values is an equivalent Boolean expression!