Lab Exercise 1

Learning Outcomes Covered

- How to process user input using the input() function in Python;
- How to format output using f-strings;
- Using arithmetic operators like +, -, *, /, //, % and **.
- Using conditional statements like **if .. else**;
- Using looping statements like **while** and **for .. in**;
- How to generate random numbers in Python.

Instructions

- Unless specified otherwise,
  - you are allowed to use anything in the Python Standard Library;
  - you are NOT allowed to use any third-party library in your code;
  - you can assume all user inputs are always valid and your program for each question need not include code to validate them.
- The underlined blue texts in the Sample Runs section of some questions refer to the user inputs. It does NOT mean that your program needs to underline them.
- **Please name each of your Python script files exactly as the name in parenthesis as shown in each question heading. Marks will be deducted if violating this instruction.**

Question 1 (q1.py) (20%)

Write a Python program to convert and print the following temperature readings (in the left column) from degree Celsius to their equivalents in degree Fahrenheit.

Degree in Celsius	Degree in Fahrenheit
0	32
100	212
16	60.8
-17.7	0.14

You can simply “hard code" the statements for each temperature conversion. The formula for the conversion is given by:

$$F = \frac{9}{5}C + 32$$

where *C* and *F* represent the temperature in degree Celsius and degree Fahrenheit respectively.

The output may have any number of decimal places for the resulted floating-point number.

Sample Output:

```
32.0
212.0
60.8
0.14000000000000057
```

Question 2 (q2.py) (20%)

Traditional vending machines often need to return the change in coins to the buyer and embed a small program to control their operation. In this question, we are to simulate that program which calculates the number of coins per face value for a specific amount of change in cents to return. The input and output will be performed via the keyboard and screen.

Write a Python program which lets the user enter an amount of change from 1 to 99 cents. The program responds by telling the user one combination of coins that equals that amount of change. We adopt the US currency system as follows:

Face value	Coin's name
1¢	penny
5¢	nickel
10¢	dime
25¢	quarter

(Note: In fact, there are less common 50¢ coins called half dollars in US, but we will ignore it.)

Sample Runs:

```
Enter an integer between 1 and 99: 87↵
One combination of coins that equals 87 cents is:
3 quarters
1 dime
0 nickel
2 pennies
```

```
Enter an integer between 1 and 99: 46↵
One combination of coins that equals 46 cents is:
1 quarter
2 dimes
0 nickel
1 penny
```

```
Enter an integer between 1 and 99: 59↵
One combination of coins that equals 59 cents is:
2 quarters
0 dime
1 nickel
4 pennies
```

The output for the first sample run was so because mapping with the above table,
3 × 25¢ + **1** × 10¢ + **0** × 5¢ + **2** × 1¢ = 87¢.

Hints:

The floor division operator (//) and the modulus operator (%) can be useful for this question.

For example,

```
87 // 25 = 3;    87 % 25 = 12
12 // 10 = 1;    12 % 10 = 2
 2 // 5  = 0;     2 % 5  = 2
 2 // 1  = 2;     2 % 1  = 0
```

As another hint, to receive user inputs from the console, use the built-in Python function input(). You should check its description in Python documentation for more details.

Other Requirements:

- Print the plural form of each denomination conditionally, i.e., when the count is greater than 1, print "quarters", "dimes", "nickels", and "pennies"; otherwise, print "quarter", "dime", "nickel", and "penny".
- Whenever a larger denomination can form the required amount, don't use a smaller one. For example, for amount 20¢, always use 2 dimes instead of 1 dime plus 2 nickels nor 4 nickels. With this rule, you should never hit the case printing the plural form "nickels".

Question 3 (q3.py) (20%)

Write a Python program to calculate the compound interest for a person who invests $2500.00 in a savings account yielding 5 percent interest. Assuming that all interest is left on deposit in the account, calculate and print the amount of money in the account at the end of each year for 10 years. Use the following formula for determining these amounts:

$$a = p(1 + r)^n$$

where

- p is the original amount invested (i.e., the principal),
- r is the annual interest rate,
- n is the number of years, and
- a is the amount on deposit at the end of the nth year

Note: You are required to use some looping statements to attempt this question instead of hard coding the output.

Sample Output:

```
Years    Deposit
    1    2625.00
    2    2756.25
    3    2894.06
    4    3038.77
    5    3190.70
    6    3350.24
    7    3517.75
    8    3693.64
    9    3878.32
   10    4072.24
```

Other Requirements:

- The Years column should be right-aligned and having a width of 5 characters (with paddings).
- The Deposit column should be formatted to show 2 decimal places.
- "Use 4 spaces as the spacing between the Year and Deposit columns."

Question 4 (q4.py) (20%)

Write a Python program to print the multiplication table for 1 through 9. Note: You are required to use some looping statements to attempt this question instead of hard coding the output.

Sample Output:

```
1 x 1 = 1
1 x 2 = 2   2 x 2 = 4
1 x 3 = 3   2 x 3 = 6   3 x 3 = 9
1 x 4 = 4   2 x 4 = 8   3 x 4 = 12  4 x 4 = 16
1 x 5 = 5   2 x 5 = 10  3 x 5 = 15  4 x 5 = 20  5 x 5 = 25
1 x 6 = 6   2 x 6 = 12  3 x 6 = 18  4 x 6 = 24  5 x 6 = 30  6 x 6 = 36
1 x 7 = 7   2 x 7 = 14  3 x 7 = 21  4 x 7 = 28  5 x 7 = 35  6 x 7 = 42  7 x 7 = 49
1 x 8 = 8   2 x 8 = 16  3 x 8 = 24  4 x 8 = 32  5 x 8 = 40  6 x 8 = 48  7 x 8 = 56  8 x 8 = 64
1 x 9 = 9   2 x 9 = 18  3 x 9 = 27  4 x 9 = 36  5 x 9 = 45  6 x 9 = 54  7 x 9 = 63  8 x 9 = 72  9 x 9 = 81
```

Hints:

- To bypass the printing of the default newline character of print(), we can specify the keyword argument end as an empty string, e.g., `print("hello", end="")`.
- Left-aligning a number can be done using an f-string with <. For example, the code:

```python
i = 1
print(f"{i:<3d}")
```

will print 1␣␣ (with two trailing spaces) to the console screen.

For example, by running the following code snippet,

```python
print(f"{1:<3d}", end=" ")
print(f"{1:<3d}")
print(f"{10:<3d}", end=" ")
print(f"{10:<3d}")
print(f"{100:<3d}", end=" ")
print(f"{100:<3d}")
```

we will have the output below:

```
1   1
10  10
100 100
```

Make sure the program output is correct and its overall format agrees with the given sample output. The format requirements are:

- The 1st line has 1 equation, the ith line has i equations, the 9th line has 9 equations.
- The marks (i.e., x and =) must be aligned vertically, as shown in the sample output.

- The multiplication result, which may have 1 or 2 digit(s), must be occupying 2 characters (with a space as padding) and left-aligned as shown above.
- Use 1 space as the spacing between the columns of the equations.

Question 5 (q5.py) (20%)

Recall that a quadratic equation is represented by $ax^2 + bx + c = 0$, where *a* ≠ 0, *b*, *c* are coefficients and *x* is the unknown of the equation.

Write a Python program that helps high school students learn to judge whether a quadratic equation has real solution(s). The program should first generate 3 random integers as the coefficients of the equation, and then prompt the user to enter an answer to tell whether the equation has any real solutions. For example, the program will ask "Does (6x^2) + (8x) + (1) has real solution(s)?" where *a* = 6, *b* = 8 and *c* = 1, which are randomly generated. The student will type "YES", "NO" or "QUIT" as input in the console, and the program will tell if it is correct or not. If correct, the message will be "**Right!**"; otherwise the message will be "**Wrong!**".

The above steps will repeat, so new questions keep showing up until the user enters the word "QUIT" which will lead to the end of the program. A message "**Bye!**" is printed before the program exits.

For simplicity, assume that the coefficients *a*, *b* and *c* are always integers in the range [-99, 99] but a cannot be 0.

Sample Output:

```
Does (28x^2)+(-85x)+(40) has real solution(s)? yes
Right!
Does (-77x^2)+(85x)+(26) has real solution(s)? No
Wrong!
Does (99x^2)+(-47x)+(-30) has real solution(s)? NO
Wrong!
Does (-22x^2)+(28x)+(86) has real solution(s)? yEs
Right!
Does (-56x^2)+(65x)+(-97) has real solution(s)? yeS
Wrong!
Does (-61x^2)+(32x)+(68) has real solution(s)? nO
Wrong!
Does (38x^2)+(16x)+(51) has real solution(s)? quIt
Bye!
```

Note: The checking of the user inputs is done in a case-insensitive manner, i.e., inputs "YES", "yes", "Yes", "yEs" and "yeS" are all regarded as the same. One simple way to ensure that is to convert every input into all uppercase (or all lowercase) letters using Python built-in string method upper() or lower(). Example usage is given as follows:

```
>>> "Yes".upper()
'YES'
>>> "quit".upper()
'QUIT'
```

```
>>> "NO".lower()
'no'
>>> "No".lower()
'no'
>>> answer = "qUiT"
>>> answer.lower()
'quit'
```

Hints:

To judge whether a quadratic equation has real solution(s), check the value of $b^2 - 4ac$.

$b^2 - 4ac \geq 0$	The equation has real solution(s).
$b^2 - 4ac < 0$	The equation does not have real solution(s).

To generate a random integer between two specified integer bounds (inclusive), you may use the `randint()` function in the `random` module of the Python standard library. Check out the following example code snippet for generating n random integers between 1 and 6 where n is entered by the user.

```
# a sample program illustrating user input and random number
# here we also show a way to import a function from a module
from random import randint
n = int(input('Please enter a number: '))
print('You entered:', n)
for i in range(n):
    # generate a random integer between two bounds (inclusive)
    print(randint(1, 6))
```

```
Please enter a number: 10↵
You entered: 10
3
1
2
3
2
5
5
3
4
6
```