Lab 3 (Loop Structure)

1a. (for loop) Write a program to display the consecutive numbers between 2 numbers. Assume that the first number will always be less than the second. Display from the first number up to and including the second number, one number per line. Input sample is as follows:

```
Enter first number: 3
Enter second number: 5
```

Output:

```
3
4
5
```

1b. Modify the program to display the sum of the consecutive numbers after the consecutive numbers.

1c. Modify program in 1b to cater for any 2 integer input sequence. If the first is smaller than the second, print from the smaller to larger. If the first is greater than the second, print from the larger down to the smaller. Use only 1 loop structure for the printing.

2a. (for loop) Write a program that reads 2 input values: a string and an integer. The program prints the string a number of times specified by the integer value. For example,

```
Input string: hello
Number of times to repeat: 3
hello
hello
hello
```

2b. Modify the program to read only a string and displays the output in the following format in the example:

```
Input string: hello
h
he
hel
hell
hello
```

The first line displays the first letter of the string, the second line displays the first two letters and so on. Use only ONE for range loop. Do not use nested loops.

2c. (while, sentinel) Modify program for 2a so that it repeatedly prompts for another string to process until the user keys in “exit” to end the program. E.g.,

```
Enter String: Python
Number of times to repeat: 3
Python
Python
Python
Enter String: program
Number of times to repeat: 2
program
program
Enter String: exit
end
```

3a. (for loop) Write a program that displays a multiplication table. Read an integer number and display the multiplication table as follows in this example run:

```
Enter number: 5
1 x 5 = 5
2 x 5 = 10
3 x 5 = 15
4 x 5 = 20
5 x 5 = 25
```

If the input is 5, the table displays 5 rows, each row for a multiple of 5.

3b. (while, sentinel, for loop) Modify program in 3a to allow for multiple input. The program ends when -1 is entered for the input. Example:

```
Enter number: 2
1 x 2 = 2
2 x 2 = 4
Enter number: 3
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
Enter number: -1
```

3c. (nested loop) Validate the input for question 3b. The input must be a number between 2 and 10 (inclusive of 2 and 10). Display "Invalid. Enter again" if the number is invalid, and continue to prompt for another number until a valid number is entered. Display the multiplication table for a valid number.

4. (while, sentinel loop) Write a program that simulates the point of sale at a supermarket checkout. The program inputs the quantity and price of items and displays price, the subtotal. Input ends when -1 is entered for quantity to indicate there is no more item to checkout. The GST and the final price are then computed. For example,

```
Enter quantity: 2
Enter unit price: 1.5
Subtotal is $3.00
Enter quantity: 4
Enter unit price: 2.25
Subtotal is $9.00
Enter quantity: -1
Total price is $12.00
GST is $0.84
Please pay $12.84
```

5. (while, sentinel loop) Write a program that displays the menu shown in the example run repeatedly until the user chooses 0 to quit. You may assume that user enters only valid numbers (0-3).

```
Menu
1. Option 1
2. Option 2
3. Option 3
0. Quit
```

```
Enter choice: 1
Option 1 selected

Menu
1. Option 1
2. Option 2
3. Option 3
0. Quit
Enter choice: 3
Option 3 selected

Menu
1. Option 1
2. Option 2
3. Option 3
0. Quit
Enter choice: 0
End of program
```

6. A Head or Tail guessing game is played by tossing a coin and allowing the user to guess either H or T.

   Write separate programs for each version of the game. The versions have increasing levels of complexity:

   a. (if…else) The program uses a random function to generate either H or T. User makes a guess, and the outcome is displayed:

      Run 1:

      ```
      Head or Tail (H or T): H
      Correct!
      ```

      Run 2:

      ```
      Head or Tail (H or T): H
      Incorrect.
      ```

      (User can key his guess in uppercase or lowercase(that is, H or h or T or t). Any letter other than H or T is considered as an incorrect guess.

   b. (for…range) The program starts by asking how many rounds the player wishes to play the game. When all the rounds are over, display the number of times the player made the correct guess.

      ```
      How many rounds to play? : 5
      Round 1: Head or Tail (H or T): H
      Correct!
      Round 2: Head or Tail (H or T): T
      Wrong!
      …
      Round 5: Head or Tail (H or T): H
      Correct!
      You guess 3 correct out of 5 rounds.
      ```

   c. (while) This version re-tosses the coin whenever the player makes a wrong guess and allow the player to keep guessing until he makes a correct guess. Display the number of tosses the player takes to make a correct guess.

      ```
      Head or Tail (H or T): H
      ```

```
Wrong!
Head or Tail (H or T): T
Wrong!
Head or Tail (H or T): H
Correct!
You got it in 3 tosses!
```

d. (nested while loop) Modify the program in part c). After the player guesses correctly, prompt the player whether he wishes to play again with this prompt: "do you want to play again? (y/n)". If yes, repeat the whole process.

```
Head or Tail (H or T): H
Wrong!
Head or Tail (H or T): T
Wrong!
Head or Tail (H or T): H
Correct!
You got it in 3 tosses!
Play again? (y/n): y
Head or Tail (H or T): H
You got it in 2 tosses!
Play again? (y/n): n
Program end
```

7a. Write a program to play a "high low" guessing game. Allow the player to specify a lower and upper limit for the number that your program will 'think' of, e.g., lower limit 1 and upper limit 100 means any number between 1 and 100, inclusive of 1 and 100. After the program has 'thought of a number, allow the player to guess the number repeatedly. Validate that the player's guess is within the lower and upper limits.

The program displays "Too high" if the number guessed is higher than the actual number and "Too low" if the number guessed is less than the actual number. The program continues until the player enters the correct guess. A example run is as follows:

```
Enter your guess: 1000
Outside limit.
Enter your guess: 20
Too high.
Enter your guess: 16
You got it in 2 tries!
```

When the number guessed is correct, display a message including the number of tries taken to get the answer.

7b. Modify the program such that after each game, the program prompts the question "Continue game?(y/n): ". If the player enters y, the game is repeated. When the player enters n, display his best attempt in all the games played. The best attempt is when he makes the least number of guesses to get the number.

Exercises on functions.

8. Modify question 1 on loops by writing a function `displayFromTo(x, y)`. The function displays consecutive numbers from the smaller to the larger number. Write also function main() for the user to enter 2 values which are used to call the function.

9. Write a function `sumSquares(num)` that has 1 integer parameter. The function returns the sum of the squares from 1 up to num. E.g. sumSquares(4) returns the sum of $1^2+2^2+3^2+4^2$. Write also function main() for the user to enter 1 value which is used to call the function. main() displays the sum.

10. Write a function `getValidValue(msg, x, y)` that prompts the user for an input using msg, between x and y (float, inclusive of both). If the value is out of range, display 'Invalid input. Try again' and prompt again for input. The function returns the valid input between x and y. Call the function using this test:

```
mark = getValidValue('Enter mark', 0, 100)
```

    Write also function main() to call the function. main() displays the mark.

11. Division of 2 numbers x by y can be done by repeatedly, subtracting y from x until y is less than x. For example, if x=11 and y=4, then 11-4-4=3. The quotient is 2 (subtract twice) and the remainder is 3. Write a function `divide(x, y)` that uses the above algorithm to return the quotient and remainder. Test the function by writing an appropriate main() function.