Introductory Computer Sciences
Problem set #4
Repetition Structures

1. Prompt the user to enter a vector.

   i. Use a *for loop* to sum the elements in the entered vector.

   ii. Use a *while loop* to sum the elements in the entered vector.

   iii. Check the answers with the *sum* function.

2. A Fibonacci sequence is composed of elements created by adding the two previous elements. The simplest Fibonacci sequence starts with 1, 1 and proceeds as follows:

   $$1, 1, 2, 3, 5, 8, 13, ...$$

   However, a Fibonacci sequence can be created with any two starting numbers. Fibonacci sequences appear regularly in nature. For example, the shell of the chambered nautilus grows in accordance with a Fibonacci sequence.

   Prompt the user to enter the first two numbers in a Fibonacci sequence and the total number of elements requested for the sequence. Find the sequence and store it in an array by using a *for loop*.

3. Repeat the preceding problem, this time using a *while loop*.

4. One interesting property of a Fibonacci sequence is that the ratio of the values of adjacent members of the sequence approaches a number called "the golden ratio". Create a program that accepts the first two numbers of a Fibonacci sequence as user input and then calculates additional values in the sequence until the ratio of adjacent values converges to within 0.001. You can do this in a while loop by comparing the ratio of element k to element $k-1$ and the ratio of element $k-1$ to element $k-2$. If you call your sequence x, then the code for the while statement is

   $$\text{while}\,\text{abs}(x(k)/x(k-1) - x(k-1)/x(k-2)) > 0.001$$

5. Edmond Halley (the astronomer famous for discovering Halley's comet) invented a fast algorithm for computing the square root of a number, A. Halley's algorithm approximates $\sqrt{A}$ as follows:

   Start with an initial guess x_1. The new approximation is then given by

   $$\begin{aligned} y_n &= \frac{1}{A}x_n^2 \\ x_{n+1} &= \frac{x_n}{8}\left(15 - y_n(10 - 3y_n)\right). \end{aligned}$$

   These two calculations are repeated until some convergence criterion, ε, is met.

   $$|x_{n+1} - x_n| \leqslant \varepsilon$$

   Write an m-file that approximates the square root of a number. It should prompt the user to enter two inputs, the initial guess and the convergence criterion. Test your code by comparing it to the value calculated with the built-in MATLAB function, *sqrt*.

```
clear
close all
clc

while(true)
    index = input('Which answer do you want to see ==> ');
    fprintf('------Answer of the %g. question------ \n',index);
    switch(index)
        case 1
            disp('-------------------(i)-------------------');
            IN_VECTOR = input('Enter a vector defined in brackets [-] ::> ');
            IN_VECTOR
            disp('-------------------(ii)------------------');
            SUM = 0;
            i = 1;
            while(i <= numel(IN_VECTOR))
                SUM = SUM + IN_VECTOR(i);
                i = i + 1;
            end
            SUM
            disp('-------------------(iii)-----------------');
            sumFunc = sum(IN_VECTOR)
            SUM
        case 2
            FIRST_ELEMENT = input('Enter the 1st starting number ::> ');
            SECOND_ELEMENT = input('Enter the 2nd starting number ::> ');
            SEQUENCE_SIZE = input('Enter the sequence length ::> ');
            FIBONACCI_SEQUENCE(SEQUENCE_SIZE) = 0;
            FIBONACCI_SEQUENCE(1) = FIRST_ELEMENT;
            FIBONACCI_SEQUENCE(2) = SECOND_ELEMENT;
            for i = 3 : SEQUENCE_SIZE
                FIBONACCI_SEQUENCE(i) = FIBONACCI_SEQUENCE(i - 1) + FIBONACCI_SEQUENCE(i - 2);
            end
            FIBONACCI_SEQUENCE
        case 3
            FIRST_ELEMENT = input('Enter the 1st starting number ::> ');
            SECOND_ELEMENT = input('Enter the 2nd starting number ::> ');
            SEQUENCE_SIZE = input('Enter the sequence length ::> ');
            FIBONACCI_SEQUENCE(SEQUENCE_SIZE) = 0;
            FIBONACCI_SEQUENCE(1) = FIRST_ELEMENT;
            FIBONACCI_SEQUENCE(2) = SECOND_ELEMENT;
            i = 3;
            while(i <= SEQUENCE_SIZE)
                FIBONACCI_SEQUENCE(i) = FIBONACCI_SEQUENCE(i - 1) + FIBONACCI_SEQUENCE(i - 2);
                i = i + 1;
            end
            FIBONACCI_SEQUENCE
        case 4
            FIRST_ELEMENT = input('Enter the 1st starting number ::> ');
            SECOND_ELEMENT = input('Enter the 2nd starting number ::> ');
            FIBONACCI_SEQUENCE(1) = FIRST_ELEMENT;
            FIBONACCI_SEQUENCE(2) = SECOND_ELEMENT;
            FIBONACCI_SEQUENCE(3) = FIBONACCI_SEQUENCE(1) + FIBONACCI_SEQUENCE(2);
            i = 3;
            while (abs(FIBONACCI_SEQUENCE(i) / FIBONACCI_SEQUENCE(i - 1) - ...
                       FIBONACCI_SEQUENCE(i - 1) / FIBONACCI_SEQUENCE(i - 2)) > 0.001)
                i = i + 1;
                FIBONACCI_SEQUENCE(i) = FIBONACCI_SEQUENCE(i - 1) + FIBONACCI_SEQUENCE(i - 2);
            end
            FIBONACCI_SEQUENCE
```

```
        case 5
            % It is up to you to take A as pre-defined value(e.g: 16)
            A = input('Enter the number whose square root to be found::> ');
            INITIAL_GUESS = input('Enter the initial guess [X0] ::> ');
            CONVERGENCE = input('Enter the convergence criterion [EPS] ::> ');
            disp('Edmond Halley Algorithm For square root');
            SQRT_OF_A = EdmondHalleySqrt(A, INITIAL_GUESS, CONVERGENCE)
            disp('MATLab built-in function sqrt(x) For square root');
            SQRT_OF_A = sqrt(A)
        otherwise
            disp('Please enter a valid value!!');
    end
    disp('****************************************************');
    control = input('Do you want to look another one?(Y/N)==> ','s');
    if(control == 'y' | control == 'Y')
        clear
        close all
        clc
        continue
    else
        clear
        close all
        clc
        disp('Bye bye..');
        break
    end
end

function OUT = EdmondHalleySqrt(A, X, EPS)
X_PREV = X;
Y = 0;
while (1)
    Y = X_PREV ^ 2 / A;
    X = (X_PREV / 8) * (15 - Y * (10 - 3 * Y));
    if (abs(X - X_PREV) <= EPS)
       OUT = X;
       break
    end
    X_PREV = X;
end
end
```

Published with MATLAB® R2019b