BME1901 – Introductory Computer Sciences
Laboratory Handout – 6

OBJECTIVES

Learn about,

- User defined functions
- factorial() function
- break statement
- Solving various questions

TOOLS

User defined functions[1]

A user defined function with syntax "[y1,...,yN] = myfun(x1,...,xM)" declares a function named "myfun" that accepts inputs "x1,...,xM" and returns outputs "y1,...,yN". This declaration statement must be the first executable line of the function. Valid function names begin with an alphabetic character, and can contain letters, numbers, or underscores. After the declaration of the function user may define any statement as the part of the function. Built-in MATLAB functions, user defined functions, for & while loops, and if-elseif-else clauses may be called within a function file. "end" may be used to mark the end of statements realted to a user defined function.

```
Function [outputs] = FunctionName(inputs)

    statements

end
```

Example: Write a function "add" that accepts two inputs, adds them together, and returns it.

```
function c = add(a,b)

    c = a + b;

end
```

```
>> add(2,3)

ans =

    5
```

Example: Write a function "average" that accepts an input vector, calculates the average of the values, and returns a single result.

```
function ave = average(x)

    ave = sum(x)/length(x);

end
```

```
>> z=1:99;
>> y = average(z)

y =
    50
```

Example: Write a function "stat" that returns the mean and standard deviation of an input vector.

```
function [m,s] = stat(x)

    n = length(x);

    m = sum(x)/n;

    s = (sum((x-m).^2/n))^(1/2);

end
```

```
>> values = [12.7, 45.4, 98.9, 26.6, 53.1];
>> [ave,stdev] = stat(values)

ave =
    47.3400

stdev =
    29.4124
```

[1] https://www.mathworks.com/help/matlab/ref/function.html

factorial() function[2]

Function factorial(n) returns the product of all positive integers less than or equal to n, where n is a nonnegative integer value. If n is an array, then f contains the factorial of each value of n. The data type and size of f is the same as that of n. The factorial of n is commonly written in math notation using the exclamation point character as n!. Note that n! is not a valid MATLAB syntax for calculating the factorial of n.

```
>> factorial(10)

ans =

    3628800
```

break statement[3]

"break" statement terminates the execution of a for or while loop. Statements in the loop after the "break" statement do not execute (statements before break execute). In nested loops, break exits only from the loop in which it occurs. Control passes to the statement that follows the end of that loop.

Example: Write a script (m-file) "sum_rand.m" which sums a sequence of uniformly distributed random numbers between 0 and 1 until the next random number is greater than a given upper limit. Then, exit the loop using a break statement.

```
limit = 0.8;

s = 0;

while 1

    tmp = rand(1);

    if tmp > limit

        break

    end

    s = s + tmp;

end

disp(s)
```

[2] https://www.mathworks.com/help/matlab/ref/factorial.html
[3] https://www.mathworks.com/help/matlab/ref/break.html

BME1901 – Introductory Computer Sciences
Laboratory Handout – 6

PROBLEMS

1. Write a function called "sort3" that accepts three number inputs, then sorts them in increasing order in a vector, and returns the vector. (You may not use any built-in MATLAB functions.)

2. Write a function called "taylor_exp" that would calculate the exponential function given below using Taylor Series approximation.

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

   Function should accept two number inputs. First is the exponent "x" and second is the upper limit of the last term in the sum (e.g.:10^{-6}). When the last term is smaller than the given limit program should end returning the approximation for e^x as the first output and how many iterations it took as the second output. (Use for loop from 1 to infinity in your solution).

3. Write a function called "guess_number" which does not have any inputs or outputs. Function should create a random integer between 0 and 10 from a uniformly distributed population. Then function should require an input from the user as their guess for the number. If the user's number is smaller or bigger then the function's number, the program should print a relevant message (Too big/small. Try again.) on the screen. This process should continue until the user guesses the number correctly or user runs out of three tries. If the user correctly guesses the number in three tries the program should end printing success with how many tries that it took. If the user runs out of three tires the program should end priting failure.

4. Write a function called "throw_dice" that accepts how many dice (6 sided dice) to be rolled randomly and returns each of their value as a vector. Then write another function called "marmut" that accepts how many players are playing. This function should roll two dice (using throw_dice function) and record the sum of dice for each player. Before each roll the program should request the user to hit enter, then roll the dice and display the result for each player. In any moment if a player rolls 10, 11 or 12 as the sum of his/her dice that player direcly wins the game and function ends printing that player as the winner with his/her result. If no one wins via rolling 10, 11 or 12 then the winner(s) is the player(s) who rolled the highest sum. Hence, function prints his/her player number(s) and result(s) before ending.