INTRODUCTION TO FUNCTIONS IN MATLAB

LAB # 03

Spring 2023

CSE301L Signals & Systems Lab

Submitted by: **Ali Asghar**

Registration No. : **21PWCSE2059**

Class Section: **C**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Sumayyea Salahuddin

Date:

March 20, 2023

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Lab Objective(s):

In this lab, we will get an understanding of the following topics:

- Making Functions
- Control Structures
- Relational Constructs
- Logical Constructs
- Branching Constructs
- Looping constructs

Task # 01:

Write a function that accepts temperature in degrees F and computes the corresponding value in degrees C. The relation between the two is

$$T_c = \frac{5}{9}(T_c - 32)$$

Be sure to test your function.

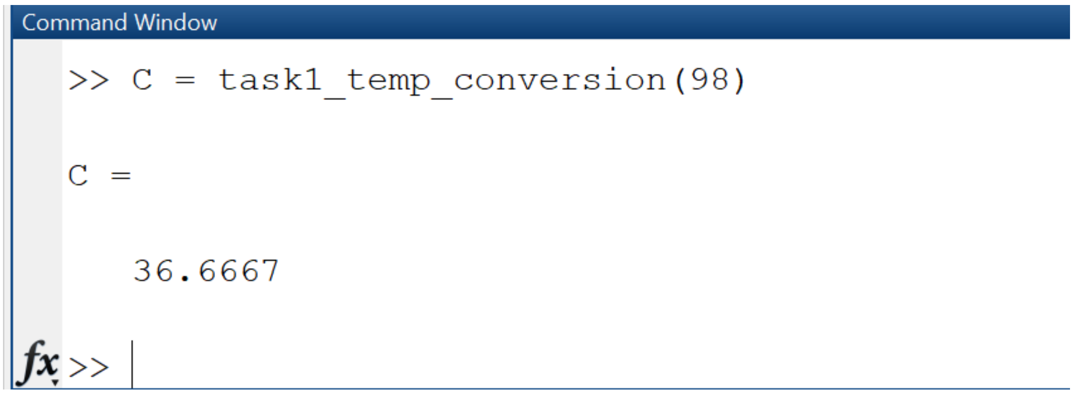
Problem Analysis:

We must convert temperature between different units in some operations and experiments. Here, we use MATLAB to convert temperature between different units.

Algorithm:

- Create function task1_temp_conversion.
- Pass temp in Fahrenheit during function call.
- Put this value in conversion formula and store it in Tc variable
- Display Tc

Output / Graphs / Plots / Results:



```
>> C = task1_temp_conversion(98)

C =

   36.6667

>> 
```

Discussion and Conclusion:

In MATLAB, it is simple to convert temperatures.

Task # 02:

For the arrays x and y given below, write MATLAB code to find all the elements in x that are greater than the corresponding elements in y.
x = [-3, 0, 0, 2, 6, 8] y = [-5, -2, 0, 3, 4, 10]

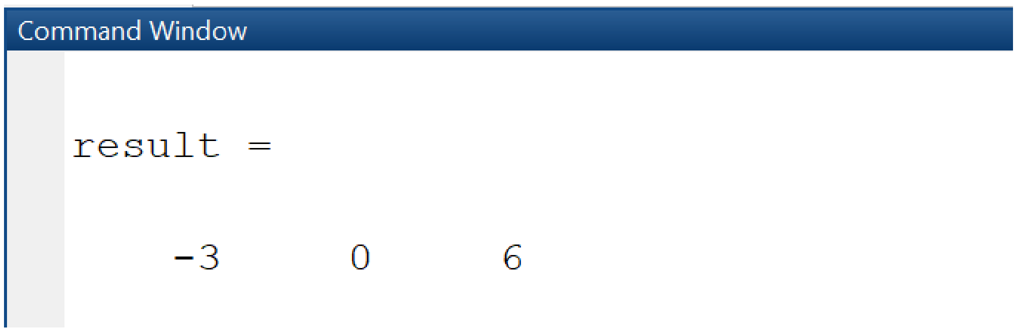
Problem Analysis:

Every complex mathematical calculation requires an array comparison. Here, we use MATLAB to do an array comparison.

Algorithm:

- Write both the arrays.
- Pass both arrays into comparison.
- Store the greater values in result.
- Display result.

Output / Graphs / Plots / Results:



```
result =

    -3      0      6
```

Discussion and Conclusion:

We have contrasted these two arrays here. Logic true or false array is the outcome. If array element 1 is greater than array element 2, the index returns true; otherwise, it returns false.

Task # 03:

For $0 < a \leq 16$, find the values of C defined as follows:

$$C = \begin{cases} 4ab & \text{for} \quad 1 \leq a \leq 8 \\ ab & \text{for} \quad 8 < a \leq 16 \end{cases}$$

and b = 12.

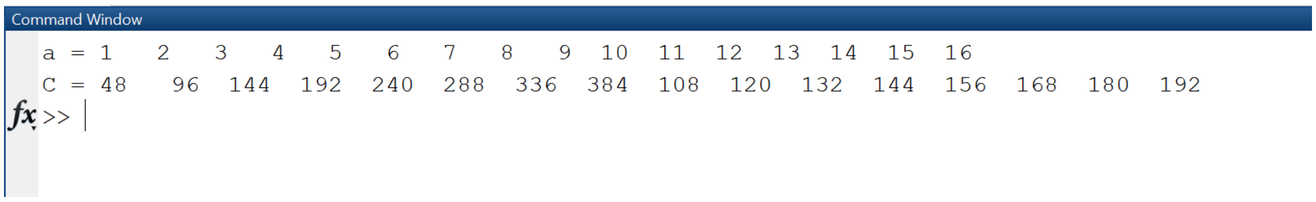
Problem Analysis:

Mathematical conditional calculations can be necessary. Here, we use MATLAB to conduct conditional calculations.

Algorithm:

- Take in the values of a and b.
- Impose if restrictions.
- Make an empty matrix of zeros of size a.
- Run a for loop from 1 to a.
- Calculate result in if restrictions.
- Display a and C.

Output / Graphs / Plots / Results:



```
a = 1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
C = 48    96   144   192   240   288   336   384   108   120   132   144   156   168   180   192
>> 
```

Discussion and Conclusion:

we can see that MATLAB makes conditional computations quite simple.

Task # 04:

For the values of integer, **a** going from 1 to 10, using separately the methods of **if syntax** and the Boolean alternative expressions, find the values of C if:

$$C = \begin{cases} a^2 & \text{for} \quad a < 3 \\ a + 3 & \text{for} \quad 3 \le a < 7 \\ A & \text{for} \quad a > 7 \end{cases}$$

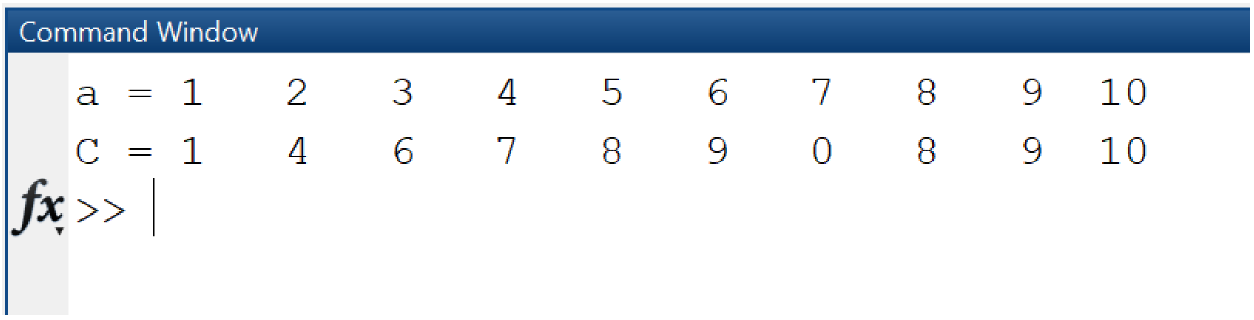
Problem Analysis:

Mathematical conditional calculations can be necessary. Here, we use MATLAB to conduct conditional calculations.

Algorithm:

- Take in the values of a and b.
- Impose if restrictions.
- Make an empty matrix of zeros of size a.
- Run a for loop from 1 to a.
- Calculate result in if restrictions.
- Display a and C.

Output / Graphs / Plots / Results:



```
a = 1     2     3     4     5     6     7     8     9    10
C = 1     4     6     7     8     9     0     8     9    10
>>
```

Discussion and Conclusion:

As a result, we can see that MATLAB makes doing conditional computations quite simple.

Task # 05:

Rewrite the following statements to use only one if statement.

```
if x < y
    if z < 5
        w = x*y*z
    end
end
```

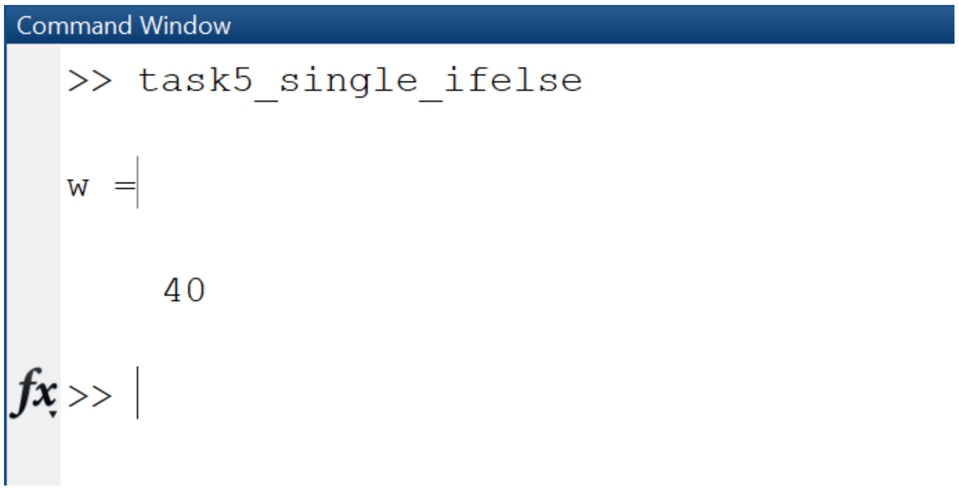
Problem Analysis:

Mathematical conditional calculations can be necessary. Here, we use MATLAB to perform conditional calculations.

Algorithm:

- Rewrite the statements
- Remove second if by using AND logic operators in first if statement

Output / Graphs / Plots / Results:


Command Window

```
>> task5_single_ifelse

w =

    40

>> 
```

Discussion and Conclusion:

As a result, we can see that MATLAB makes doing conditional computations quite simple.

Task # 06:

Using for loop, generate the cube of the first ten integers.

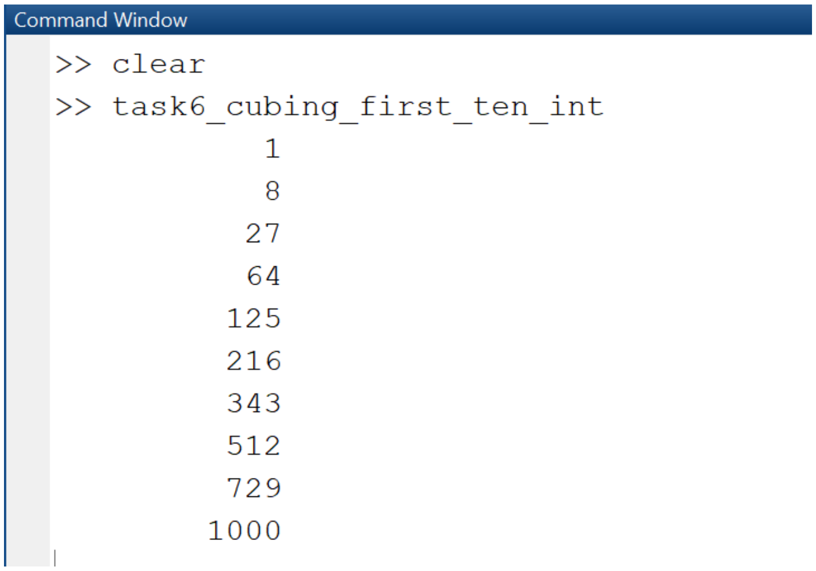
Problem Analysis:

For iterations, loops are essential in everyday programming. In MATLAB, loop iteration is done.

Algorithm:

- Write for loop for 1-10
- Store cube of iterative in cubes array
- Display cubes

Output / Graphs / Plots / Results:



```
>> clear
>> task6_cubing_first_ten_int
              1
              8
             27
             64
            125
            216
            343
            512
            729
           1000
```

Discussion and Conclusion:

We successfully iterated some variables in MATLAB using loops.

Task # 07:

Add the following two matrices using for loop.

$$A = \begin{bmatrix} 5 & 12 & 3 \\ 9 & 6 & 5 \\ 2 & 2 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 1 & 9 \\ 10 & 5 & 6 \\ 3 & 4 & 2 \end{bmatrix}$$

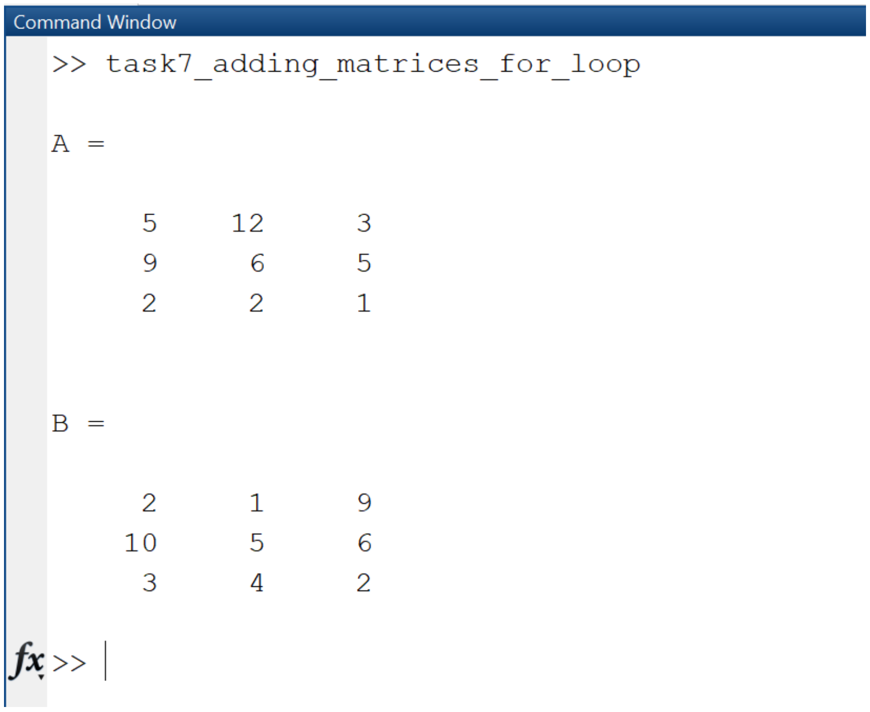
Problem Analysis:

With MATLAB, we attempt to iterate through matrix values, which is also helpful in several computing contexts.

Algorithm:

- Write both matrices A & B.
- Iterate both matrix elements into addition.
- Store all elements into a new matrix C.
- Display new matrix C.

Output / Graphs / Plots / Results:



```
>> task7_adding_matrices_for_loop

A =

     5    12     3
     9     6     5
     2     2     1


B =

     2     1     9
    10     5     6
     3     4     2

>> 
```

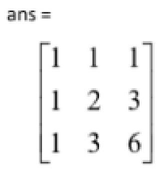
Discussion and Conclusion:

Hence, using MATLAB, we can iterate over matrices.

Task # 08:

Write MATLAB function that creates a special square matrix that has ones in the first row and first column, and whose remaining elements are the sum of two elements i.e. the element above and the element to the left, if the sum is less than 20. Otherwise, the element is the maximum of those two element values. Name the function specmat with single input defining the matrix dimension. A sample program run is:

>>specmat(3)



```
ans =
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{bmatrix}$$

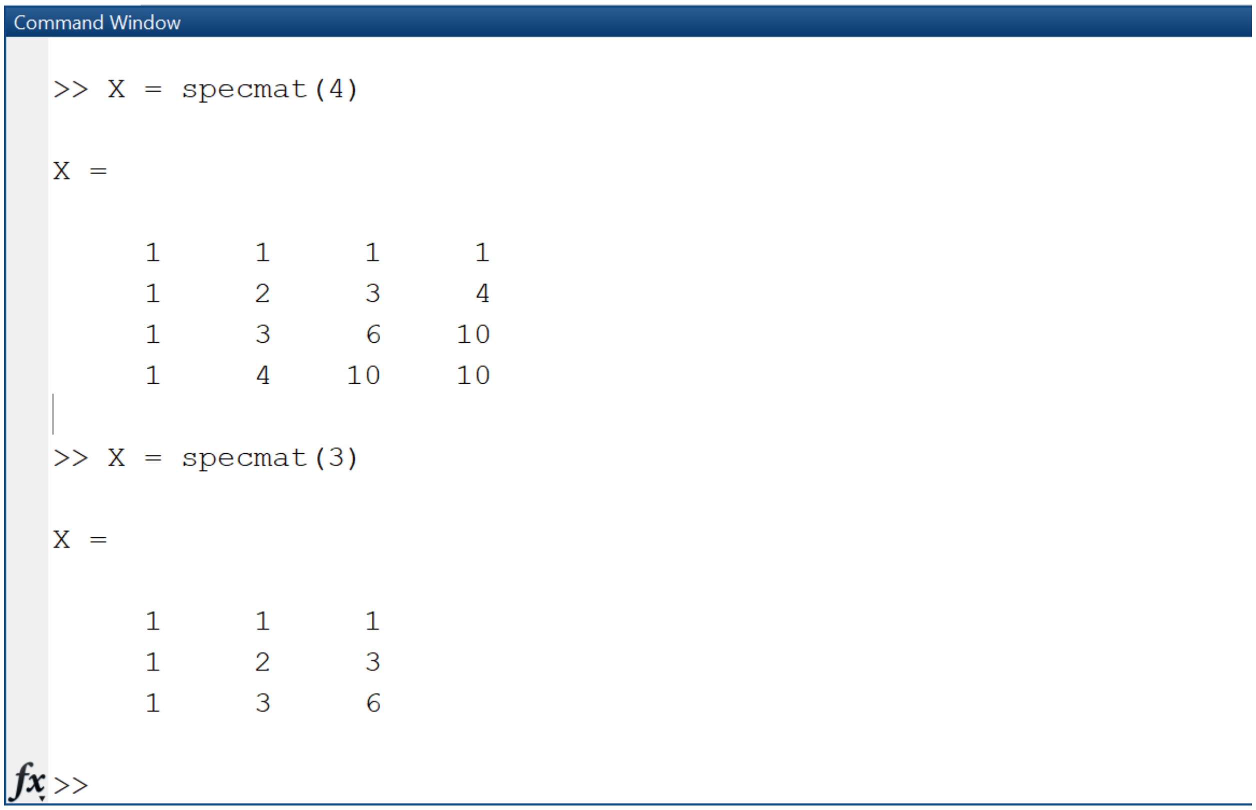
Problem Analysis:

For our objectives, we frequently need specific matrices. In MATLAB, we create unique matrices.

Algorithm:

- Make function specmat
- Write the required logic in it
- Pass the values in it

Output / Graphs / Plots / Results:



```
>> X = specmat(4)

X =

     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    10

>> X = specmat(3)

X =

     1     1     1
     1     2     3
     1     3     6

>>
```

Discussion and Conclusion:

As a result, we may use MATLAB's built-in instructions to generate matrices with the values we want.

Task # 09:

Consider the following script file. Fill in the lines of the following table with the values that would be displayed immediately after the while statement if you ran the script file. Write in the values the variables have each time the while statement is executed. You might need more or fewer lines in the table. Then type in the file, and run it to check your answers.

```
k = 1; b = -2; x = -1; y = -2;
while k <= 3
      k, b, x, y
      y = x^2 -3;
      if y < b
         b = y;
      end
      x = x + 1;
      k = k + 1;
end
```

Pass	k	b	x	y
First				
Second				
Third				
Fourth				
Fifth				

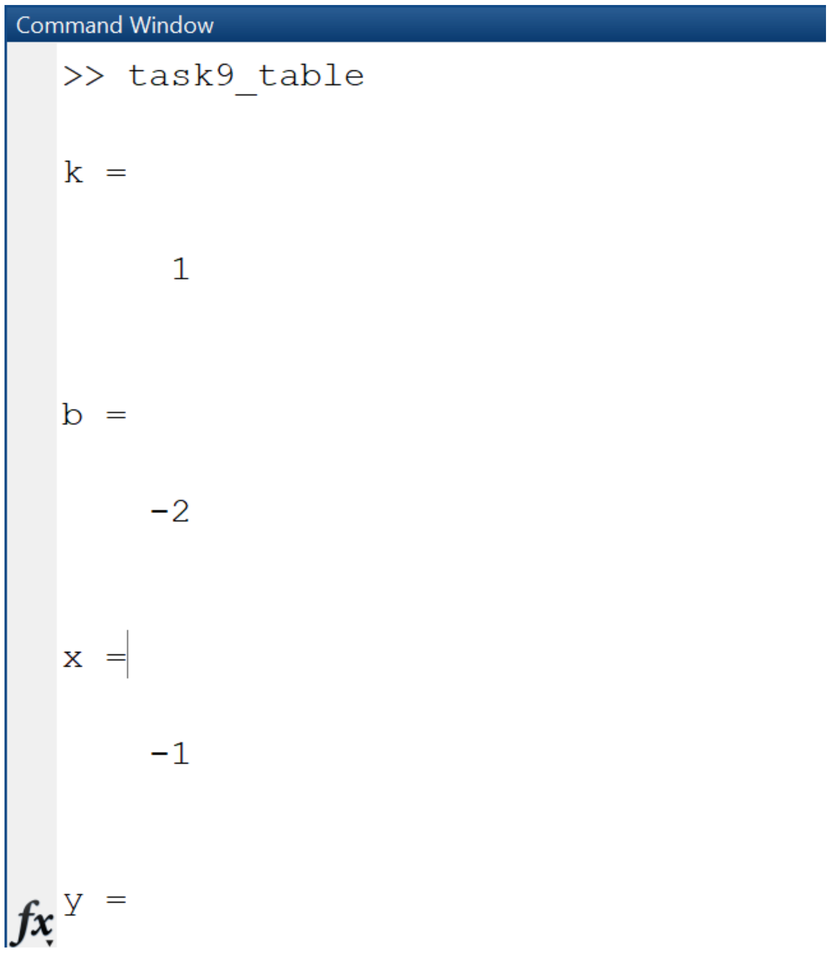
Problem Analysis:

In MATLAB, we can carry out several iterative passes.

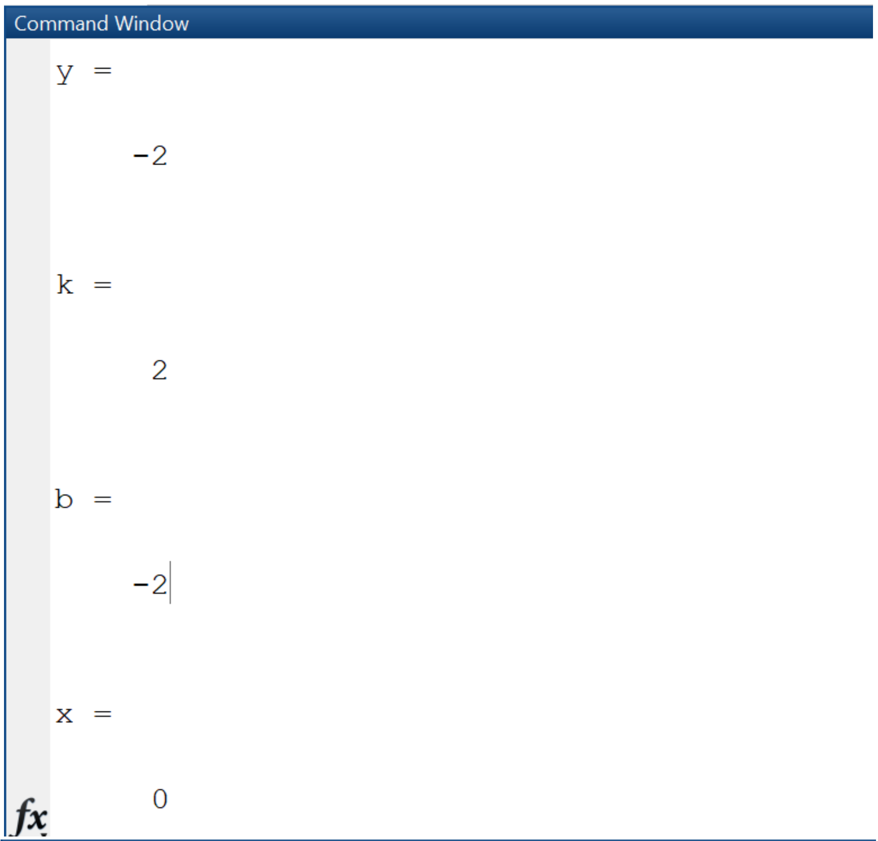
Algorithm:

- Write the code.
- Execute the code.
- Note observations.

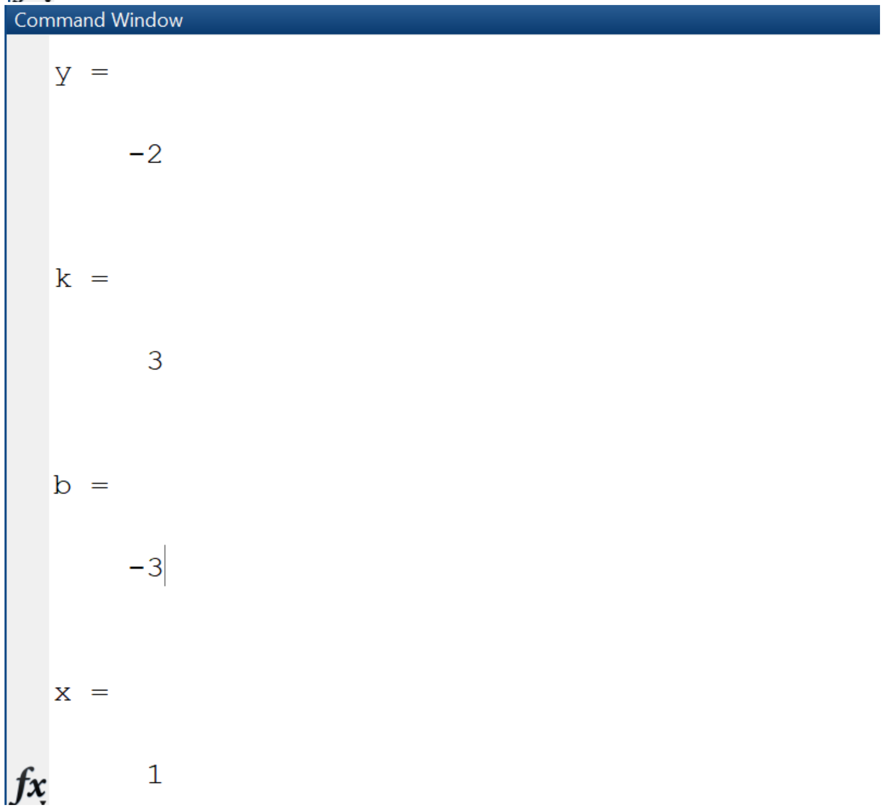
Output / Graphs / Plots / Results:


Command Window

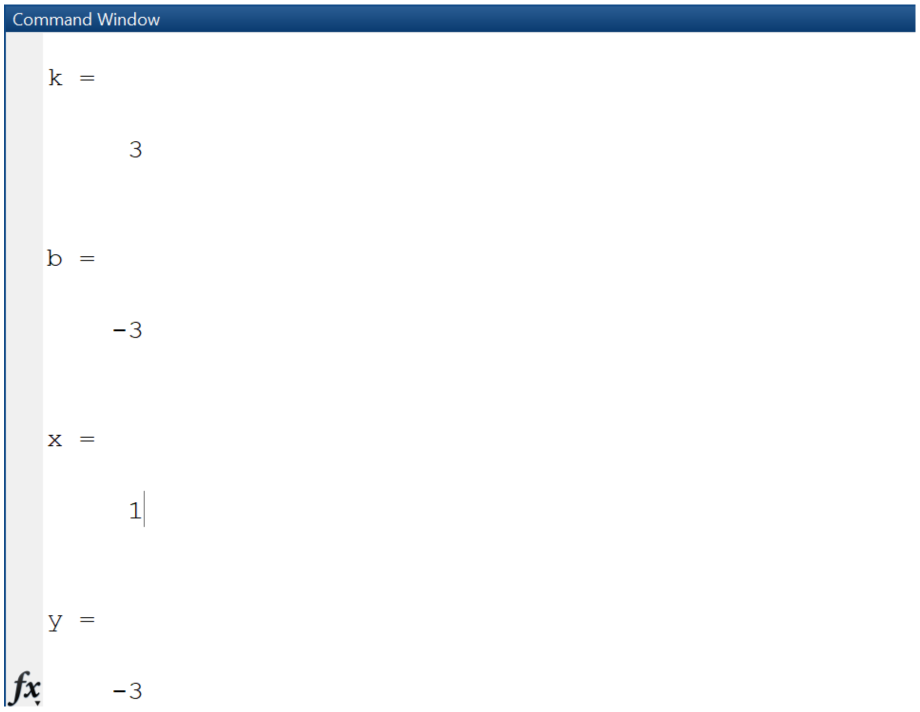
```
>> task9_table

k =

     1


b =

    -2


x =

    -1


y =
```



```
y =

    -2


k =

     2


b =

    -2


x =

     0
```



```
y =

    -2


k =

     3


b =

    -3


x =

     1
```



```
k =

     3


b =

    -3


x =

     1


y =

    -3
```

Discussion and Conclusion:

Pass	k	b	x	y
First	1	-2	-1	-2
Second	2	-2	0	-2
Third	3	-3	1	-3
Fourth	XX	XX	XX	XX
Fifth	XX	XX	XX	XX

Task # 10:

Create an m-file that inputs a number from user and then finds out the factorial of that number.

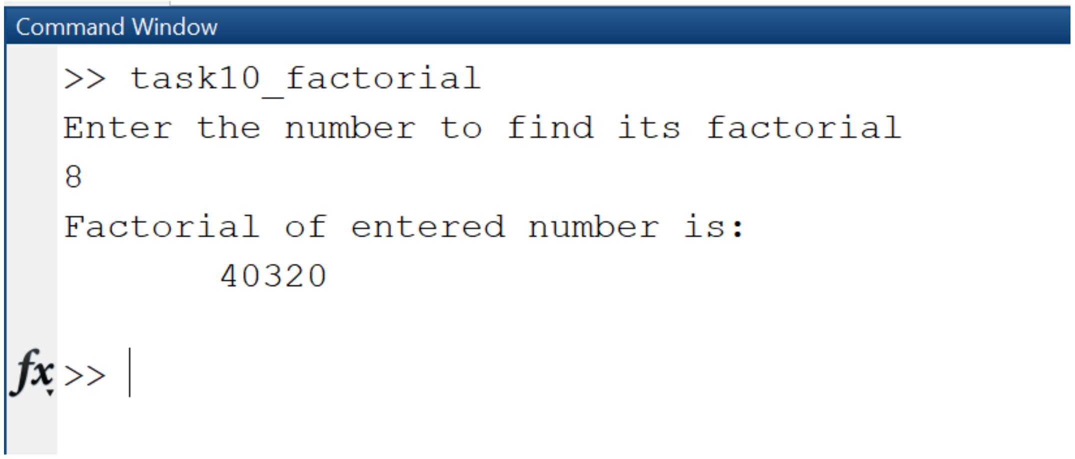
Problem Analysis:

To determine the factorial, we must apply MATLAB logic.

Algorithm:

- Input n from user
- Write factorial logic
- Pass n into fact logic and store it's factorial in fact.
- Display fact.

Output / Graphs / Plots / Results:



```
>> task10_factorial
Enter the number to find its factorial
8
Factorial of entered number is:
       40320

>> 
```

Discussion and Conclusion:

As a result, MATLAB makes finding factorial very quick.

Task # 11:

Create an m-file that takes two vectors from user. Make sure that the second vector taken is of the same size as the first vector (Hint: use while loop). In a while loop, generate a third vector that contains the sum of the squares of corresponding entries of both the vectors.

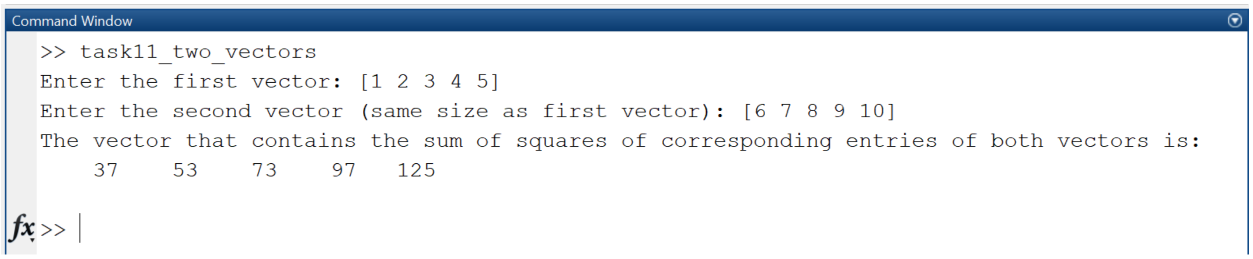
Problem Analysis:

We frequently need to combine two arrays into a single array. We perform that with MATLAB here.

Algorithm:

- Prompt the user to enter the first vector, vec1, using the 'input' function.
- Create an empty vector named vec2.
- Start a while loop that checks if the length of vec2 is not equal to the length of vec1. This loop will run until vec2 has the same length as vec1.
- Inside the while loop, prompt the user to enter the second vector, vec2, using the 'input' function. Keep prompting until vec2 has the same length as vec1.
- End the while loop when vec2 has the same length as vec1.
- Create a new vector named vec3 that will hold the sum of squares of corresponding entries of vec1 and vec2.
- Use element-wise squaring and addition to calculate the sum of squares of corresponding entries of vec1 and vec2. Store the result in vec3.
- Display a message to the user that shows vec3 using the 'disp' function.

Output / Graphs / Plots / Results:



```
>> task11_two_vectors
Enter the first vector: [1 2 3 4 5]
Enter the second vector (same size as first vector): [6 7 8 9 10]
The vector that contains the sum of squares of corresponding entries of both vectors is:
    37    53    73    97   125

>>
```

Discussion and Conclusion:

Two arrays can be iterated into a single array.

Task # 12:

Perform the following commands on various matrices and comment on each.
>> ~A
>> A&B
>> A & ~B
>> A | B

Pass k b x y
First
Second
Third
Fourth
Fifth

12

>> A | ~A

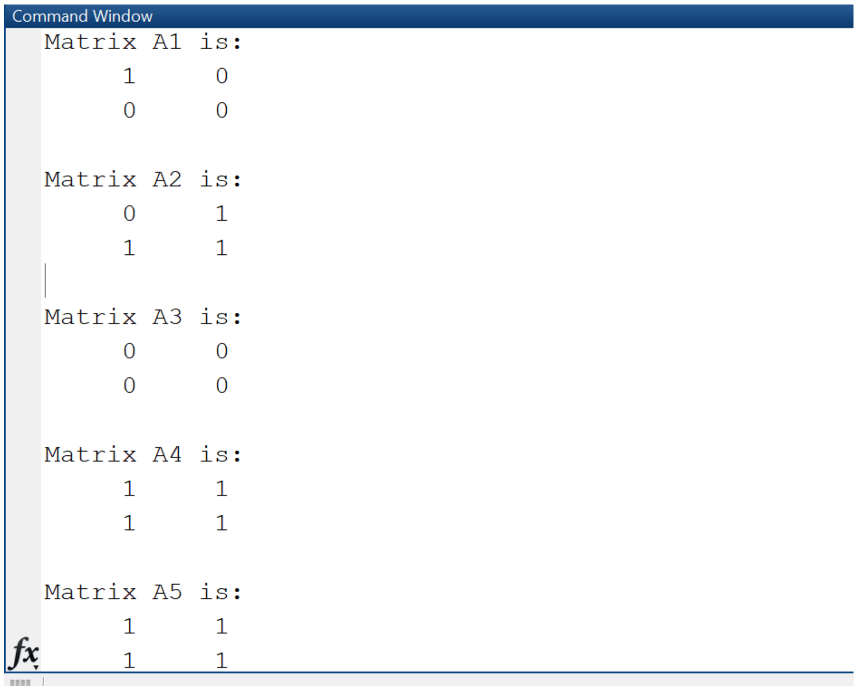
Problem Analysis:

Perform matrix operations.

Algorithm:

- Write the code
- Execute the code
- Record Observations

Output / Graphs / Plots / Results:



```
Matrix A1 is:
     1     0
     0     0

Matrix A2 is:
     0     1
     1     1

Matrix A3 is:
     0     0
     0     0

Matrix A4 is:
     1     1
     1     1

Matrix A5 is:
     1     1
     1     1
```

Discussion and Conclusion:

- In the first section, we apply a logical NOT.
- In the second section, we apply logic AND.
- In the third section, we apply logical AND and NOT.
- In the fourth section, we apply logical OR.
- In the fifth section, we apply logical OR and NOT.

Task # 13:

Design a function Fib(N) that takes N as an input and generates a Fibonacci sequence for N. Fibonacci sequence is a tile of squares whose side lengths are successive or each number is the sum of the previous number.

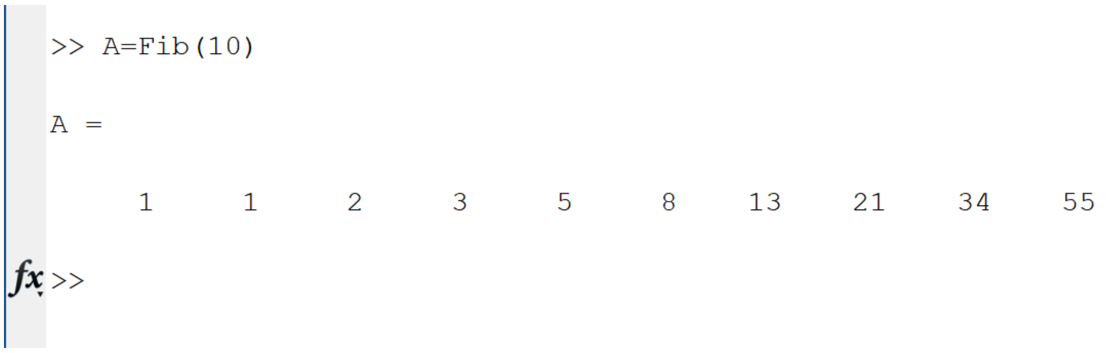
Problem Analysis:

Using MATLAB, we need to create a fibonacci sequence.

Algorithm:

- Take in the number of terms to display in Fibonacci series
- Write Fibonacci series logic and algorithm
- Display series

Output / Graphs / Plots / Results:



```
>> A=Fib(10)

A =

     1     1     2     3     5     8    13    21    34    55

>>
```

Discussion and Conclusion:

We produced a fibonacci series using MATLAB.

Task # 14:

Write a user-defined MatLab function "Calculate", with two input and two output arguments that determines the height in centimeters (cm) and mass in kilograms (kg) of a person from his height in inches (in.) and weight in pounds (lb).

(a) Determine the height and mass of a 5 ft. 15 in. person in SI units who weight 180 lb.

(b) Determine your own height and weight in SI units.

Problem Analysis:

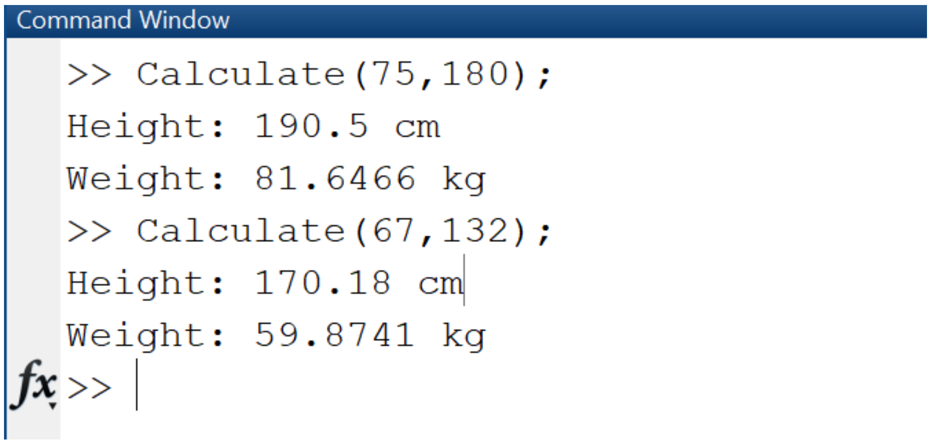
We frequently need to convert between SI units, and MATLAB makes this task simple.

Algorithm:

- Create a function Calculate with input arguments as well as return values
- Calculate pounds into kgs and inches into cm in this function

`

Output / Graphs / Plots / Results:



```
>> Calculate(75,180);
Height: 190.5 cm
Weight: 81.6466 kg
>> Calculate(67,132);
Height: 170.18 cm
Weight: 59.8741 kg
>>
```

Discussion and Conclusion:

Hence, MATLAB makes it simple to convert between SI and non-SI units.

Task # 15:

File handling in MatLab. Create files of different formats in MatLab. Use the following commands to create a text file using MatLab commands.

1. Open a file using fopen
op = fopen('weekdays.txt','wt');

2. Write the output using fprintf
fprintf(op,'Sunday\nMonday\nTuesday\nWednesday\n');
fprintf(op,'Thursday\nFriday\nSaturday\n');

3. Close the file using fclose
fclose(op);

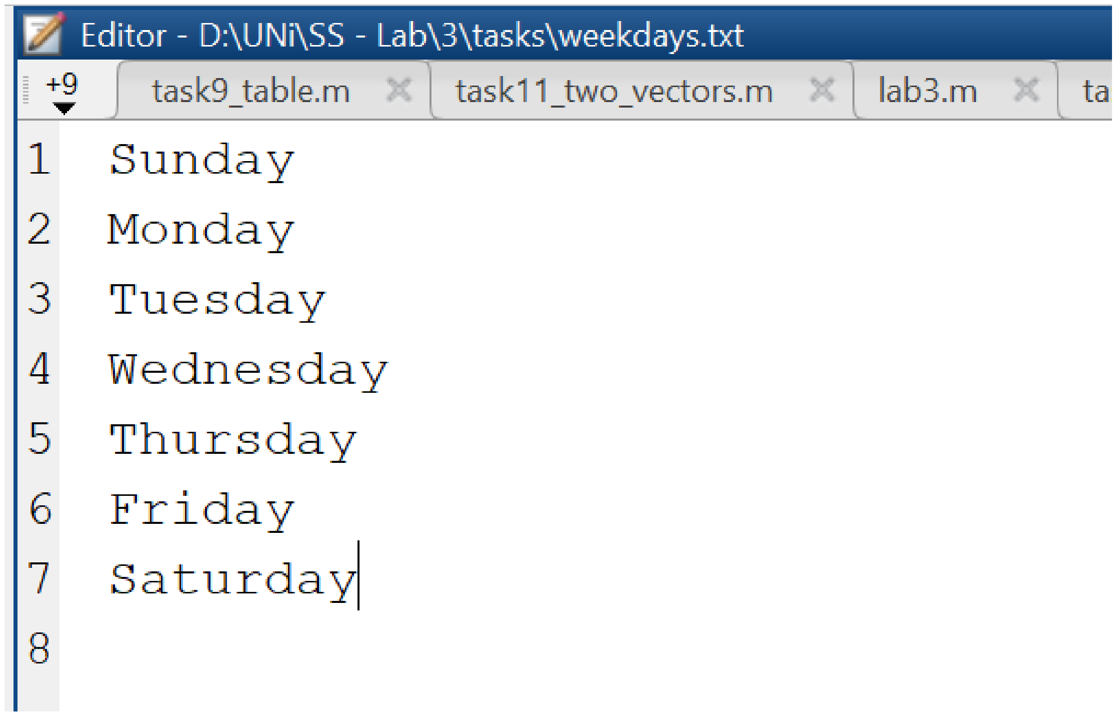
Problem Analysis:

To reduce time complexity, file management is frequently done within programmes. MATLAB is used for file handling.

Algorithm:

- Write the code
- Execute the code

Output / Graphs / Plots / Results:



```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

```

Discussion and Conclusion:

Hence, managing files in MATLAB saves us time and effort.

Task # 16:

Implement any Sorting and Searching algorithm of your choice by creating user-defined functions in MATLAB.

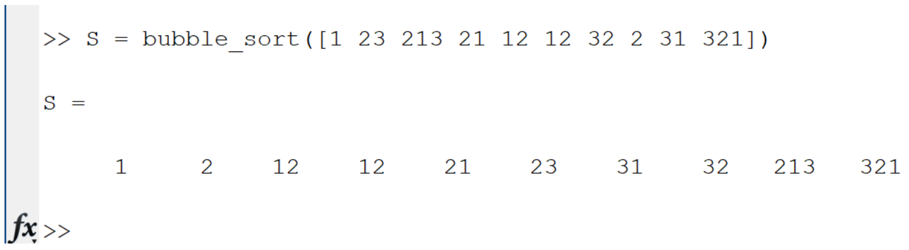
Problem Analysis:

For our comparison calculations and several operations involving data structures, we frequently require sorted data. Sorting is done via MATLAB.

Algorithm:

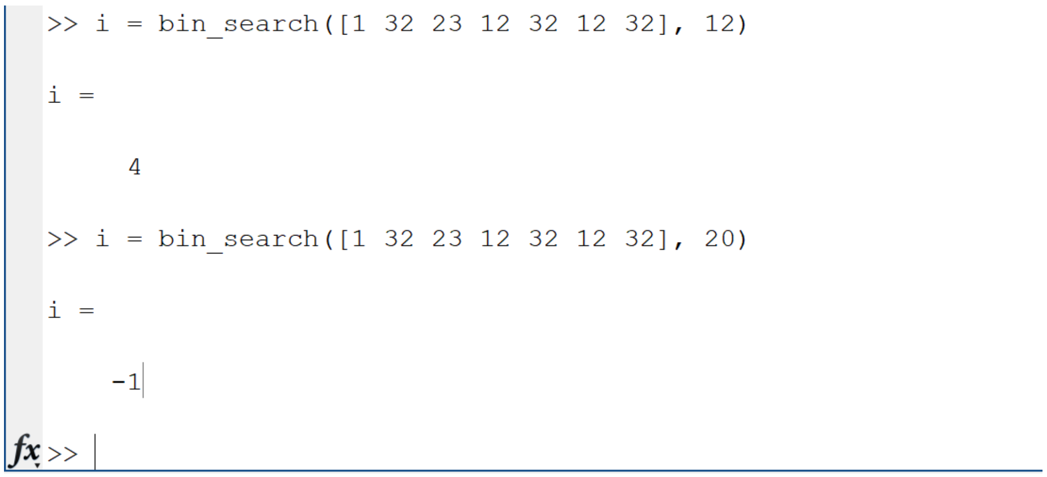
- Type in array
- Type in sorting logic
- Pass in array into the logic
- Display the result

Output / Graphs / Plots / Results:



```
>> S = bubble_sort([1 23 213 21 12 12 32 2 31 321])

S =

     1     2    12    12    21    23    31    32   213   321

fx >>
```

Bubble Sort



```
>> i = bin_search([1 32 23 12 32 12 32], 12)

i =

     4

>> i = bin_search([1 32 23 12 32 12 32], 20)

i =

    -1

fx >>
```

Binary Search

Discussion and Conclusion:

Hence, by using bubble sort and binary search, we were able to quickly sort an unsorted array and perform searching operation.