INTRODUCTION TO ALGORITHM

EEE321

Matlab Project

Beyza Sayracı

170403034

Volkan Kılıç

MATLAB FINAL PROJECT

We will examine the matlab part of my algorithm lesson project homework. A login screen [Figure 1] welcomes us. There is a gif on our login screen.

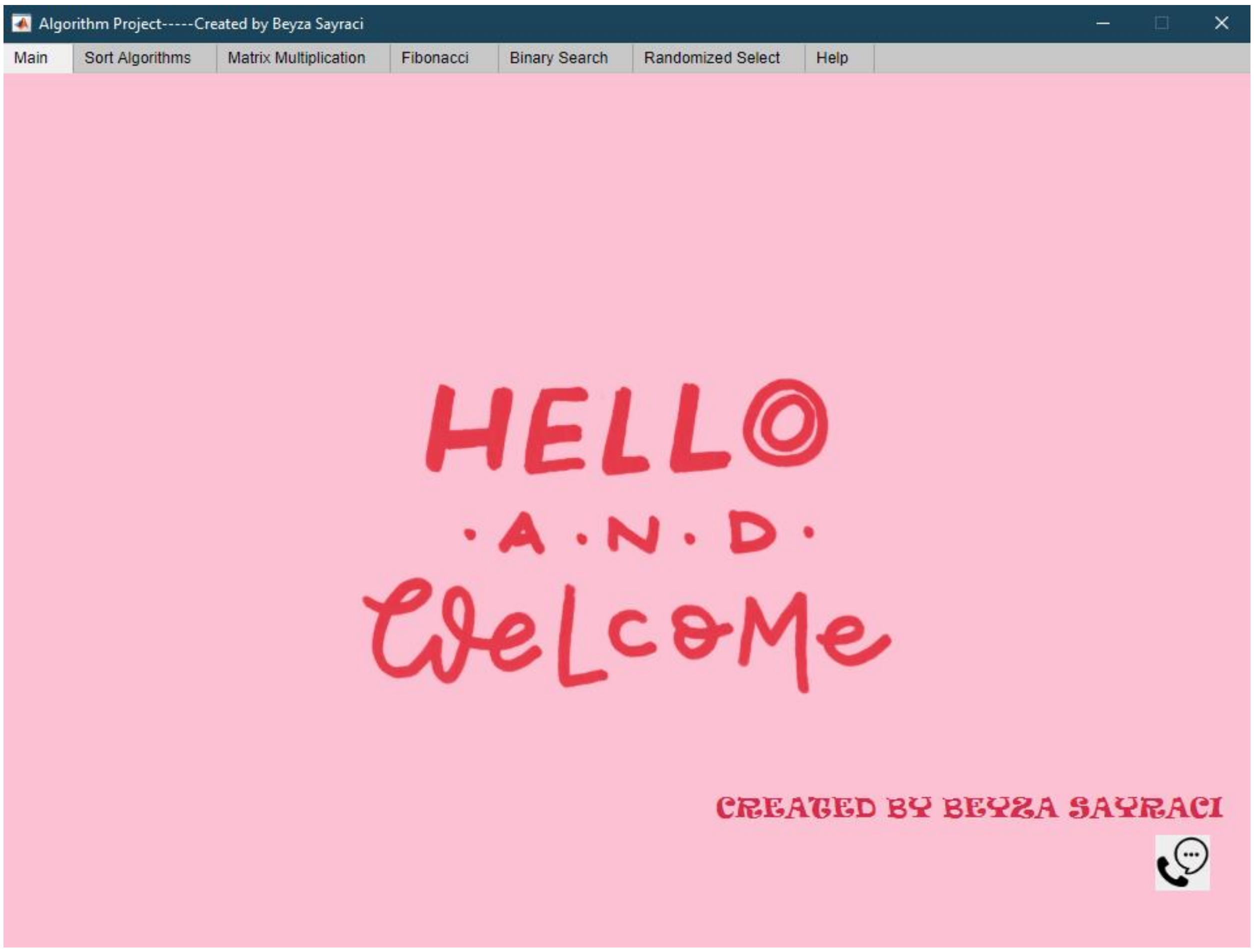


Figure 1

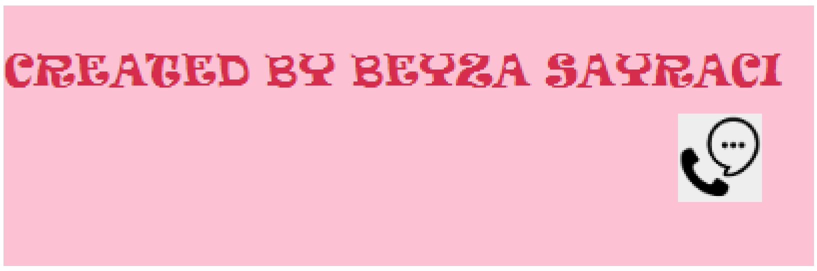
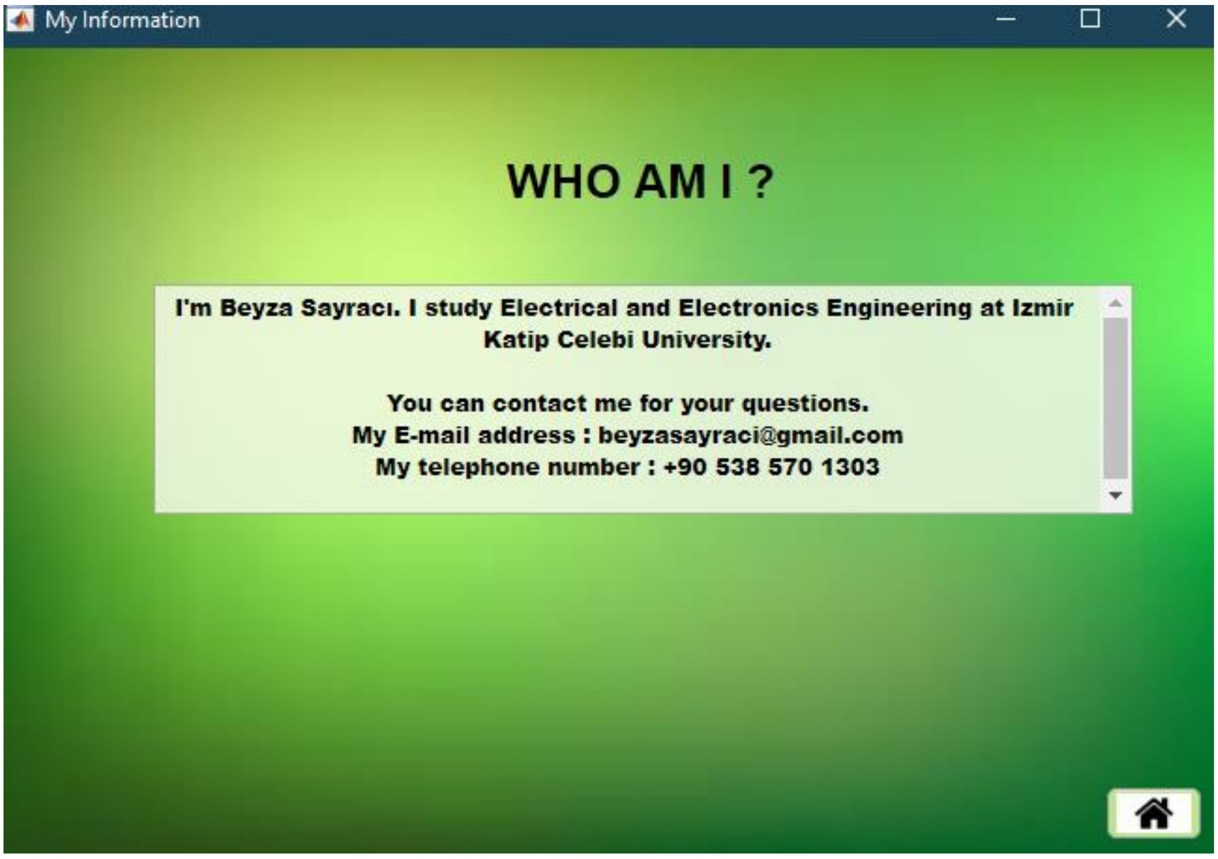


Figure 2

When we click the button in Figure 2, it opens a separate window for us. I have personal information in this window [Figure 3]. When we click on the home button, we return to the home page.


My Information

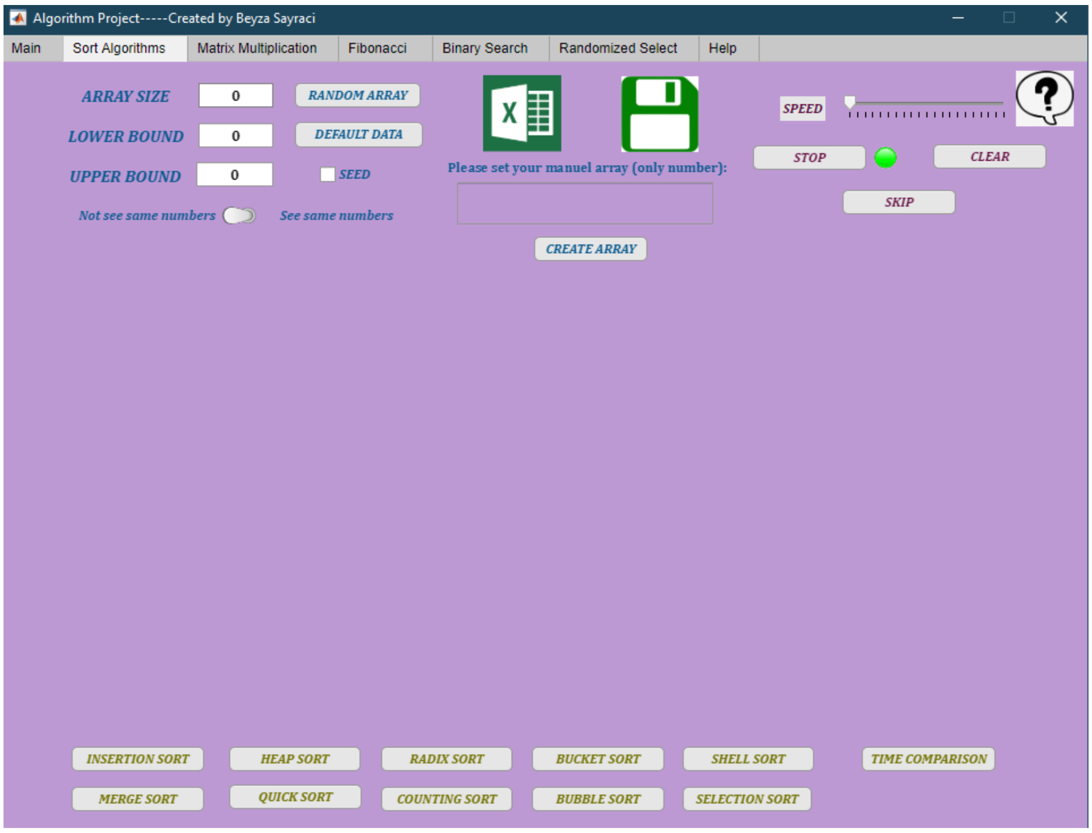
WHO AM I ?

I'm Beyza Sayracı. I study Electrical and Electronics Engineering at Izmir Katip Celebi University.

You can contact me for your questions.
My E-mail address : beyzasayraci@gmail.com
My telephone number : +90 538 570 1303

Figure 3

When we come to our second tab, sorting algorithms meet us [Figure 4].


Algorithm Project-----Created by Beyza Sayraci

ARRAY SIZE 0
LOWER BOUND 0
UPPER BOUND 0

Please set your manuel array (only number):

Figure 4

There are various sorting algorithms here. I added 2 more. These are selection sort and shell sort [Figure 5].

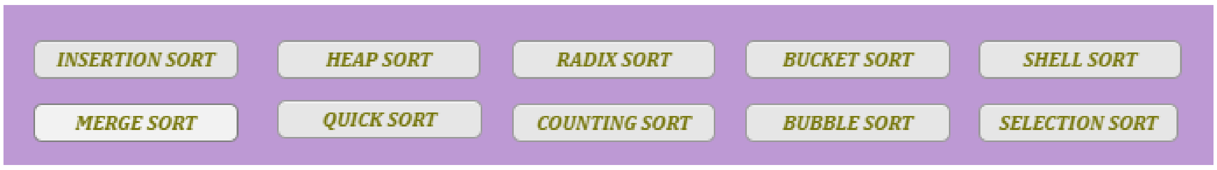


Figure 5

We can create an unsorted array in 4 different ways : by entering manually, by default, randomly and by txt, csv and excel file [Figure 6].

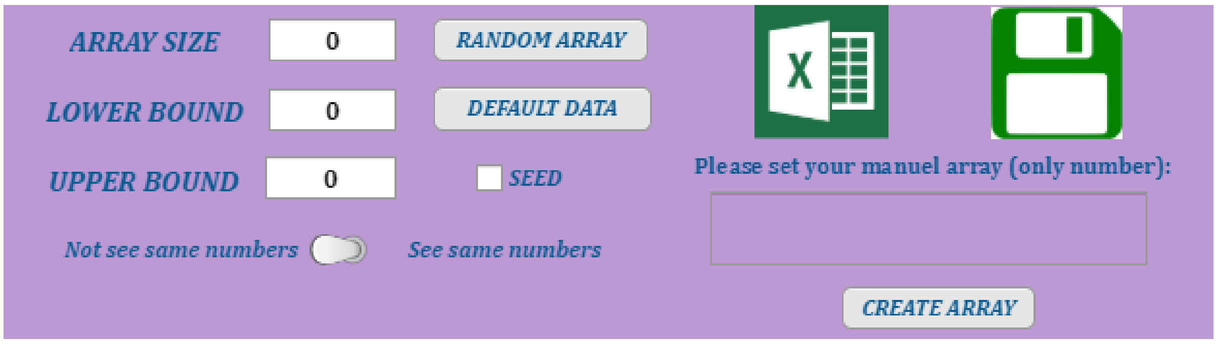
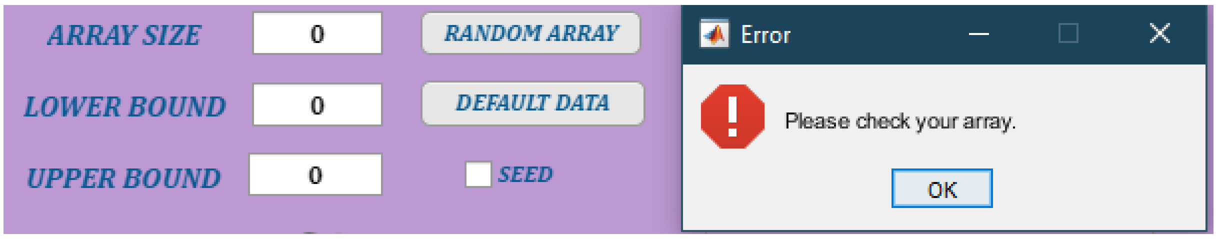


Figure 6

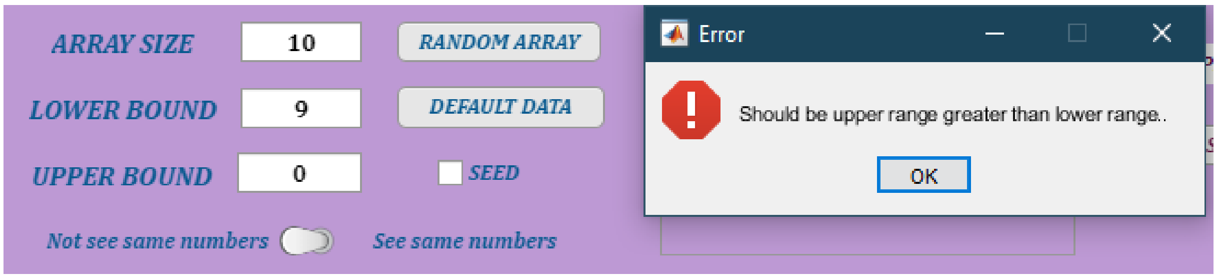
First, let's examine our random array button. There are three options we have to set : array size, upper bound and lower bound. Opening error message if they are equal to zero [Figure 7].



Please check your array.

Figure 7

We have the following option for the random array button. The option of having the same numbers or not the same numbers. If we don't want the same numbers, we turn the switch there. We let's choose see same numbers . If upper bound not greater than lower bound, open the error message box pops up [Figure 7].


ARRAY SIZE 10

LOWER BOUND 9

UPPER BOUND 0

SEED

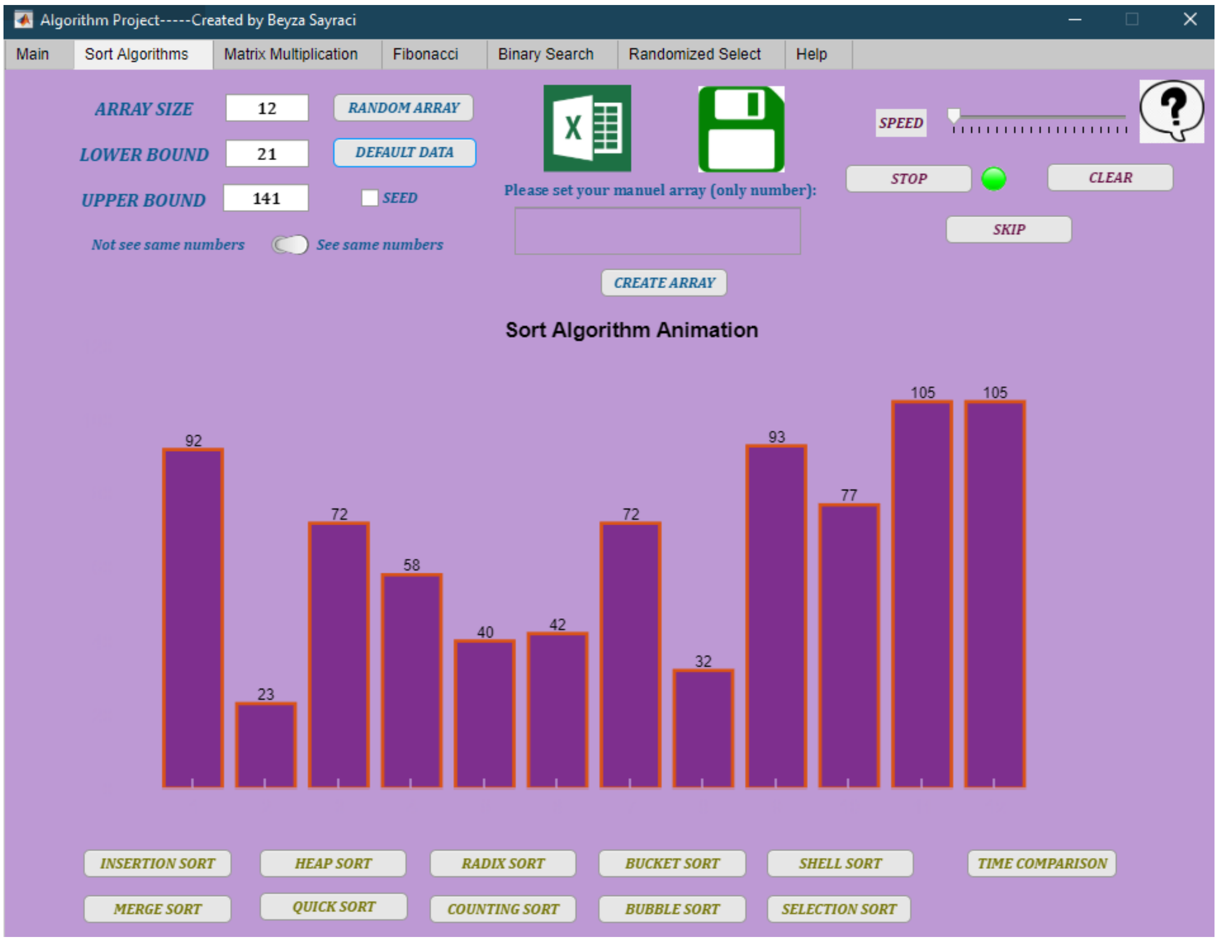
Not see same numbers See same numbers

Error

Should be upper range greater than lower range.

Figure 8

When array size, lower bound, and upper bound are appropriate, an array is formed [Figure 9].


Algorithm Project-----Created by Beyza Sayraci

ARRAY SIZE 12

LOWER BOUND 21

UPPER BOUND 141

SEED

Not see same numbers See same numbers

Please set your manuel array (only number):

Sort Algorithm Animation

92 23 72 58 40 42 72 32 93 77 105 105

Figure 9

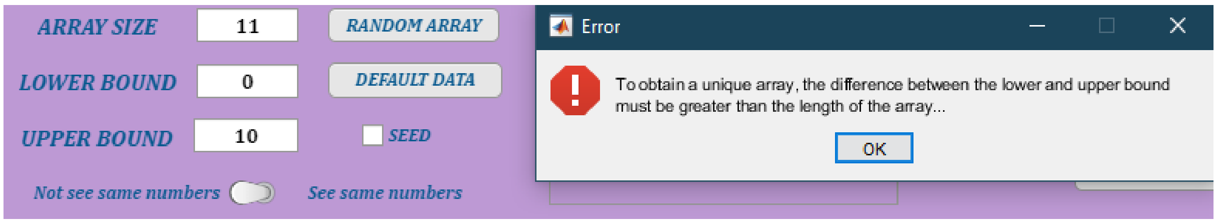
If we select the seed option this saves our array. When you enter the same array size, upper bound, lower bound values on another computer and select the see same numbers option, you will see the same array.

For the seed option, I used the **rng function** in the code.

```
rng('default');
rng(1721);
```

Now, we let's choose not see same numbers.

Another error is attached to us here. To obtain a unique array, the difference between upper bound and lower bound must be greater than array size. If you don't follow this rule, an error message appears [Figure 10]



To obtain a unique array, the difference between the lower and upper bound must be greater than the length of the array...

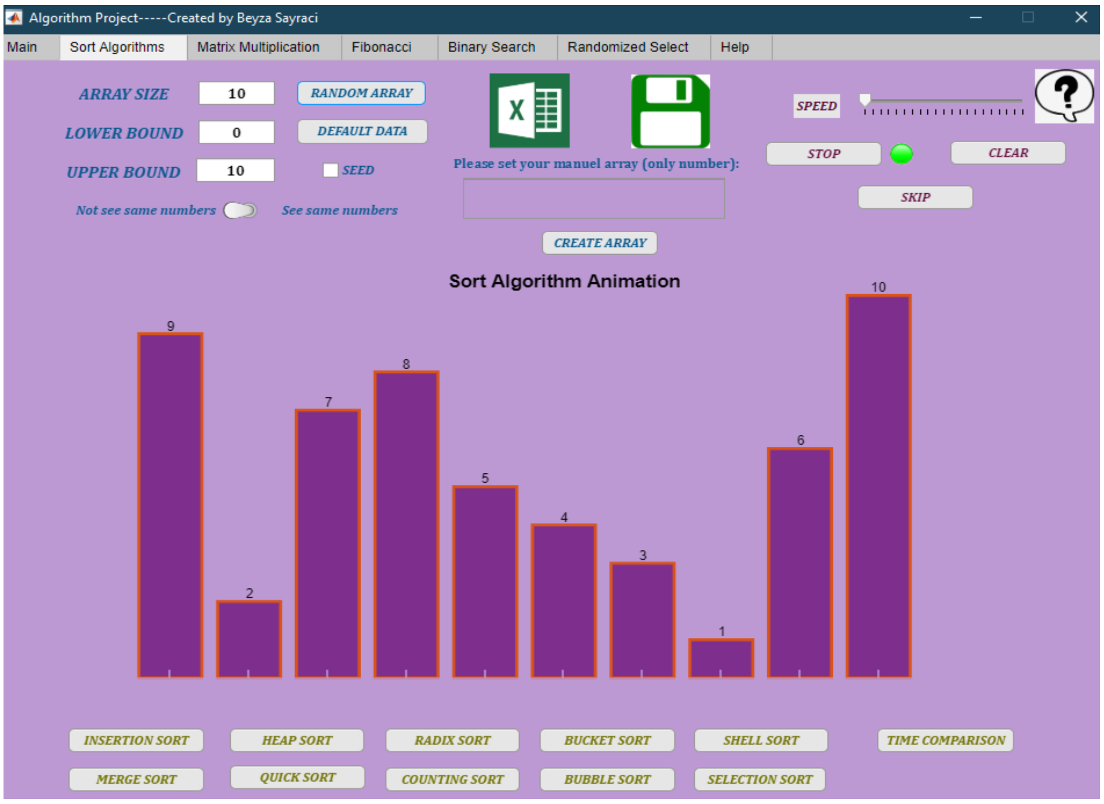
Figure 10

If I show you how I make **unique array** in our code:

```
if strcmp(same,'Not see same numbers')
    if (abs(value2-value3)<value1)
        msgbox('To obtain a unique array, the difference between the lower and upper bound mu
    else
        app.unsorted_array=[];
        while 1
            app.unsorted_array=[app.unsorted_array randi([value3 value2],1,1)];
            app.unsorted_array = unique(app.unsorted_array);

            if length(app.unsorted_array) == value1
                break
            end
        end
        app.unsorted_array = app.unsorted_array(randperm(length(app.unsorted_array)));
    end
```

I used **randi, randperm and unique function**. Let's create an array according the rules now:



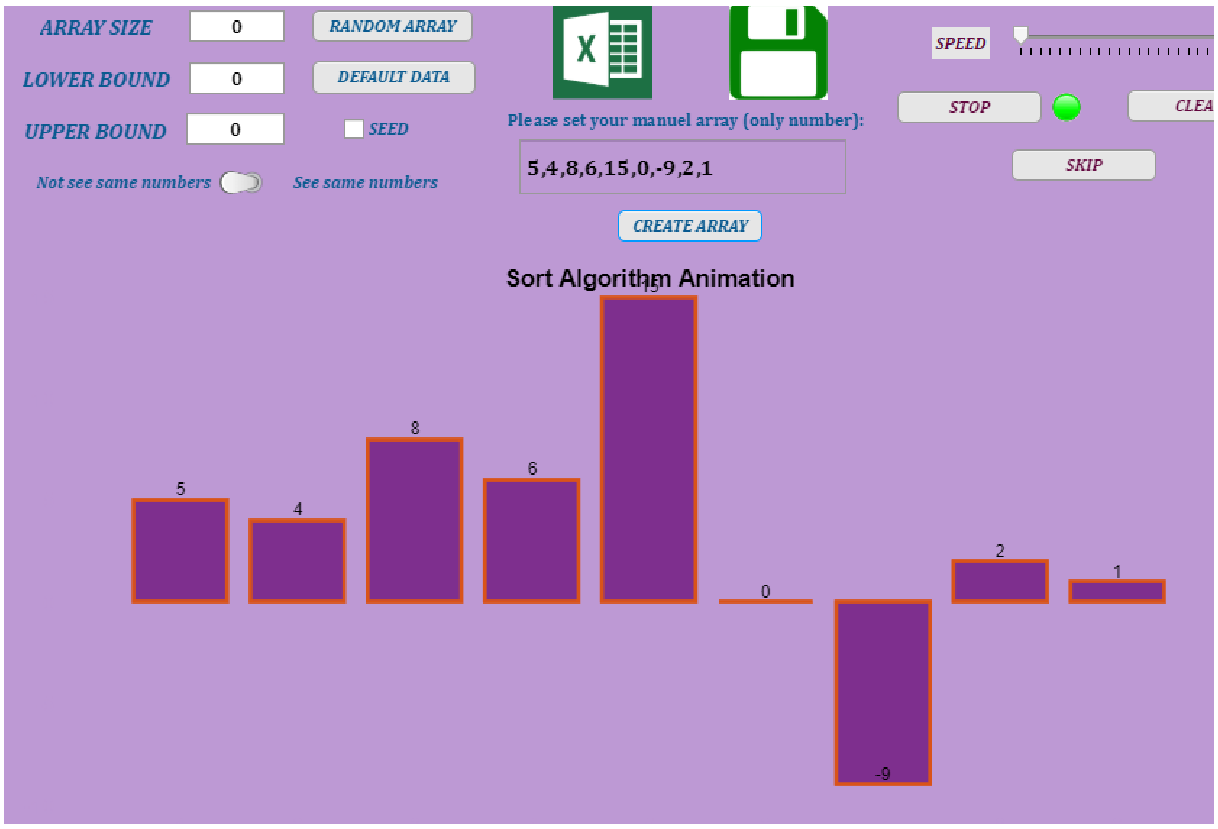
Now let's examine the default array button. With this button, an array is created without us doing anything. The seed and unique array rules in the random array are also valid in the default array. It is integrated into the code in the same way. Here I created arrays like this:

```
if strcmp(same,'Not see same numbers')
        app.unsorted_array = randperm(app.upper_bound.Value,app.array_size.Value);

else
    app.unsorted_array=randi([app.lower_bound.Value app.upper_bound.Value],1,app.array_size.Value);
end
```

In the default array, as in the random array, when we click the seed button, you save the array, and when you enter the same array size, upper bound and lower bound values on another computer, you view the same array.

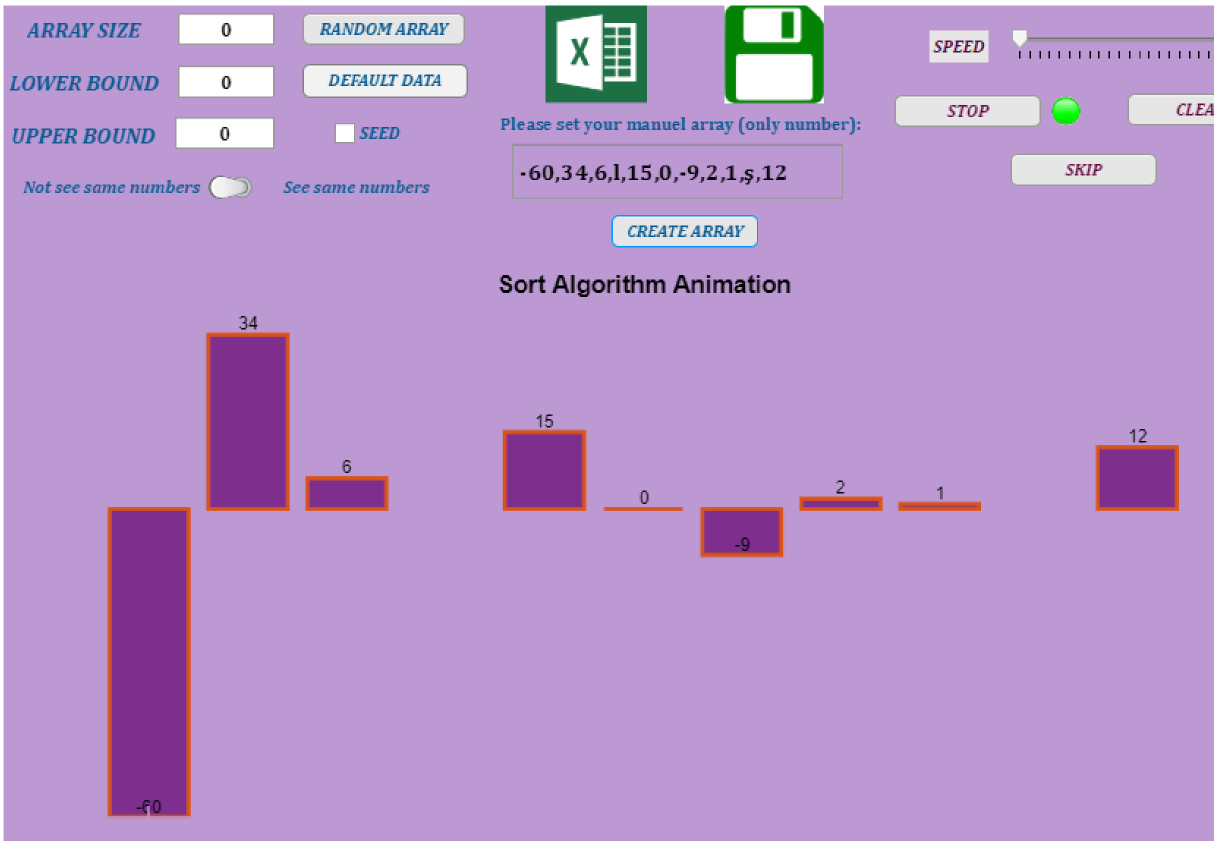
Let's come to the Create array button. When we click the button after entering our numbers here, our array is formed [Figure 11]. If we enter string, array doesn't see it and skips. So it only prints integer values[Figure 12].



Please set your manuel array (only number):

5,4,8,6,15,0,-9,2,1

Figure 11



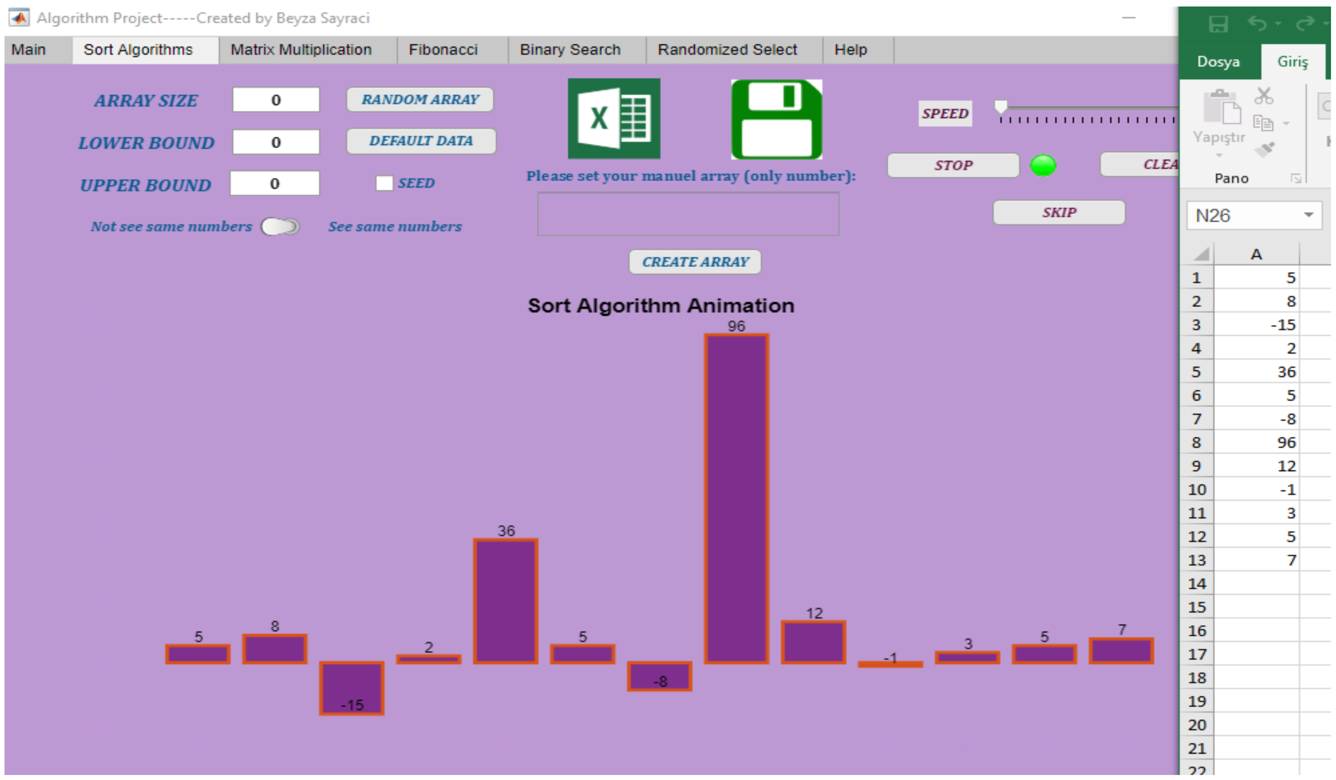
ARRAY SIZE: 0
LOWER BOUND: 0
UPPER BOUND: 0

Please set your manuel array (only number):

-60,34,6,l,15,0,-9,2,1,ş,12

Figure 12

Now let's examine getting and saving arrays from txt, csv and xlsx files.



ARRAY SIZE: 0
LOWER BOUND: 0
UPPER BOUND: 0

Please set your manuel array (only number):

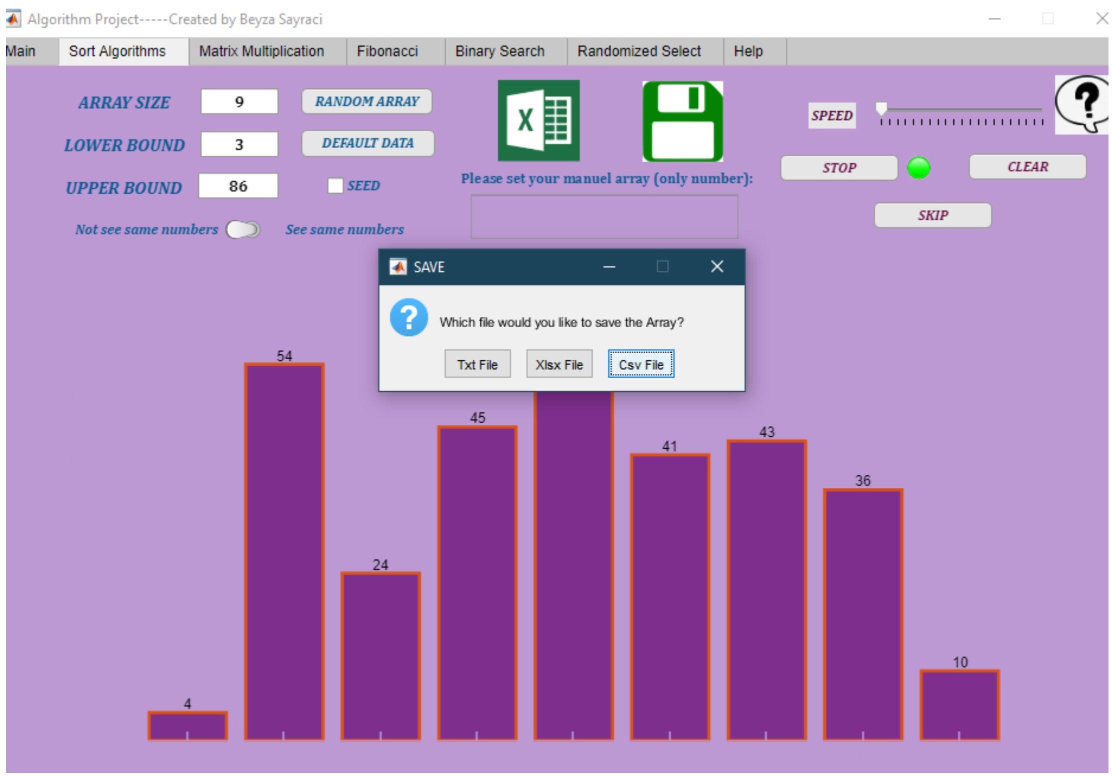
	A
1	5
2	8
3	-15
4	2
5	36
6	5
7	-8
8	96
9	12
10	-1
11	3
12	5
13	7
14	
15	
16	
17	
18	
19	
20	
21	
22	

Figure 13

As in Figure 13, I can print my excel array to bars in my interface. For this, I use the **xlsread function** in my code:

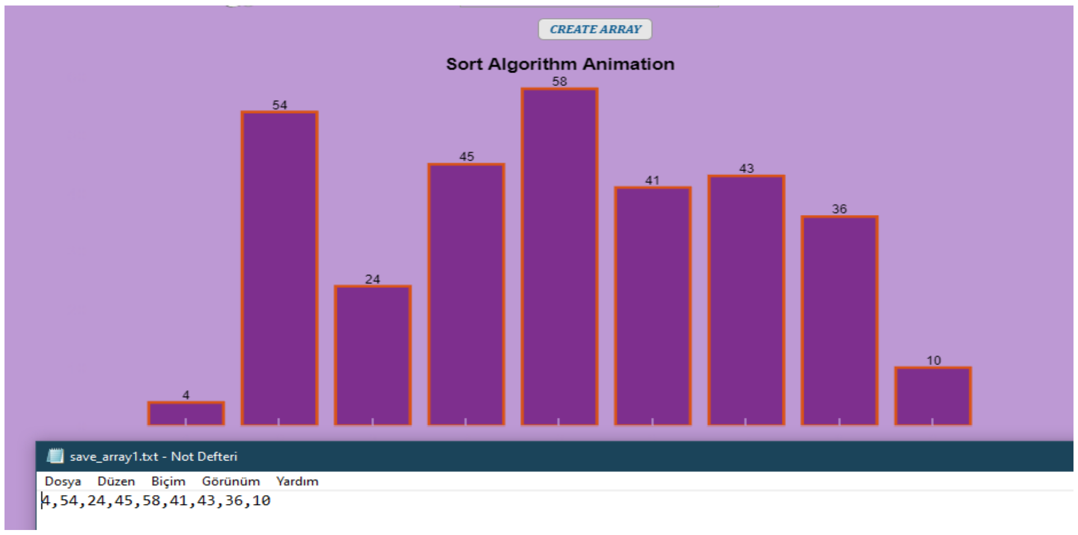
```
[file_name,PathName] = uigetfile({'*.xls;*.xlsx;'},'Please select excel file');
FileName=[PathName file_name];
app.unsorted_array=xlsread(FileName,1,'A:A');
```

On the other hand, We can save our sorted or unsorted array in three different ways.



Which file would you like to save the Array?

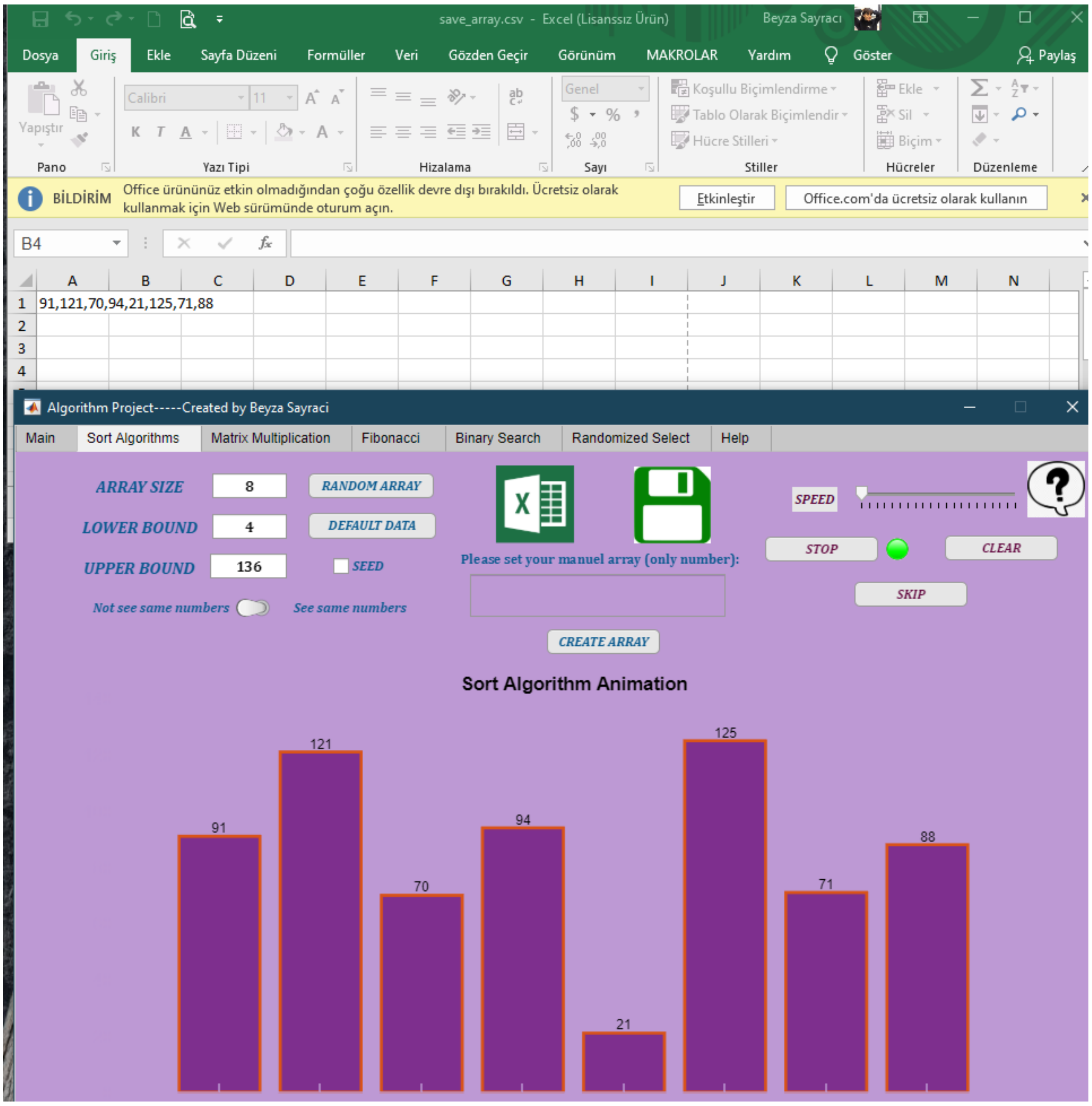
In Figure 14, I saved it as unsorted array txt file.



```
4,54,24,45,58,41,43,36,10
```

Figure 14

In Figure 15, I saved the unsorted array as a csv file.


save_array.csv - Excel (Lisanssız Ürün)

BİLDİRİM Office ürününüz etkin olmadığından çoğu özellik devre dışı bırakıldı. Ücretsiz olarak kullanmak için Web sürümünde oturum açın.

B4

	A
1	91,121,70,94,21,125,71,88
2	
3	
4	

Algorithm Project-----Created by Beyza Sayraci

ARRAY SIZE: 8
LOWER BOUND: 4
UPPER BOUND: 136

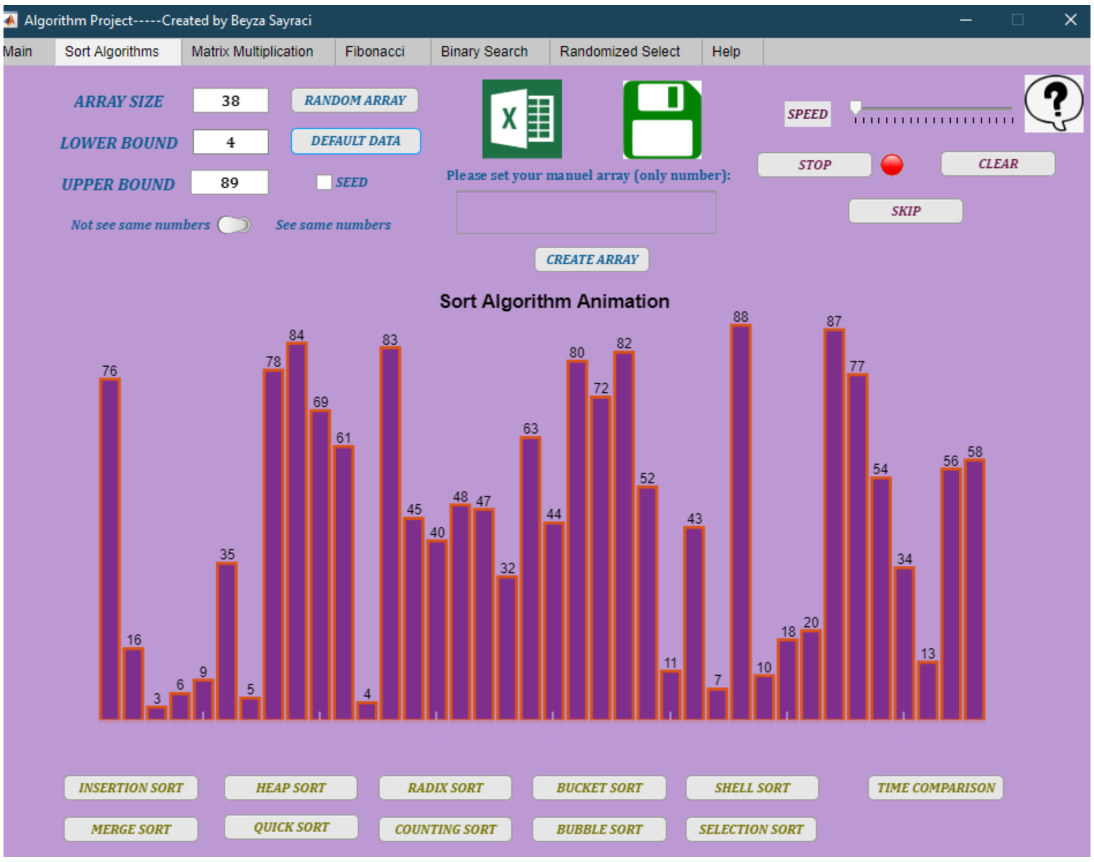
Please set your manuel array (only number):

Sort Algorithm Animation

Figure 15

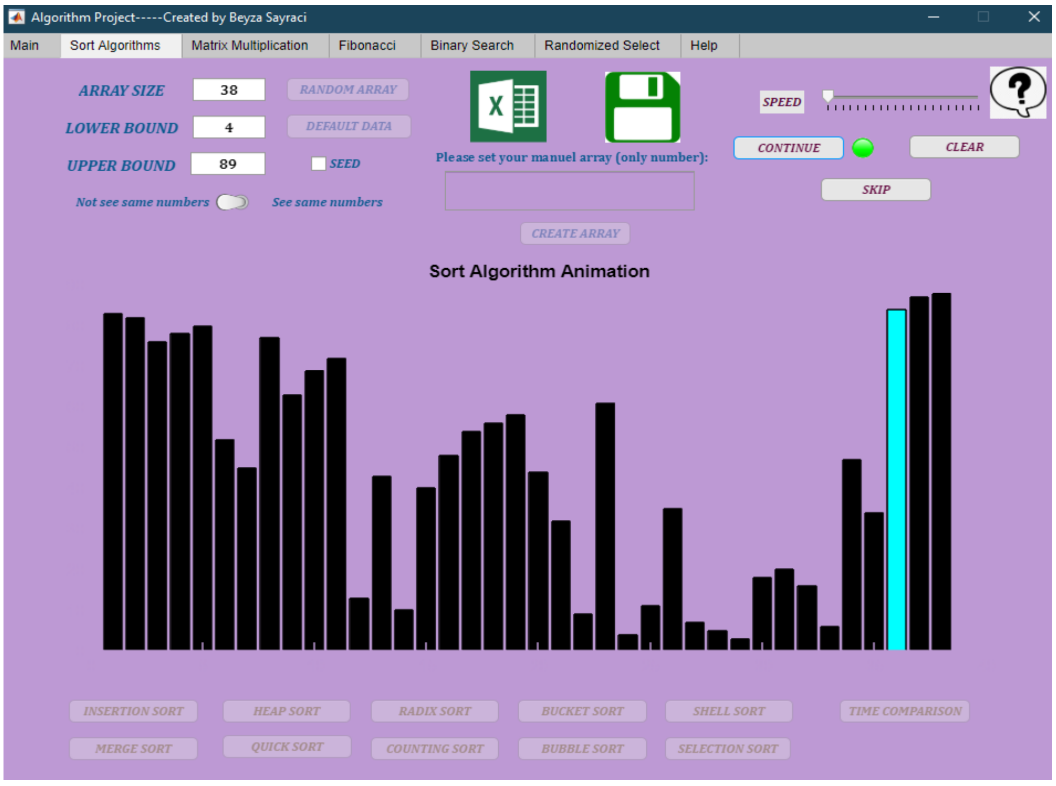
Now let's see what our **skip, stop and continue buttons** do by sorting any of our arrays.

Unsorted Array ;



Sorted Array with Heap Sort;

1-Stop Button:

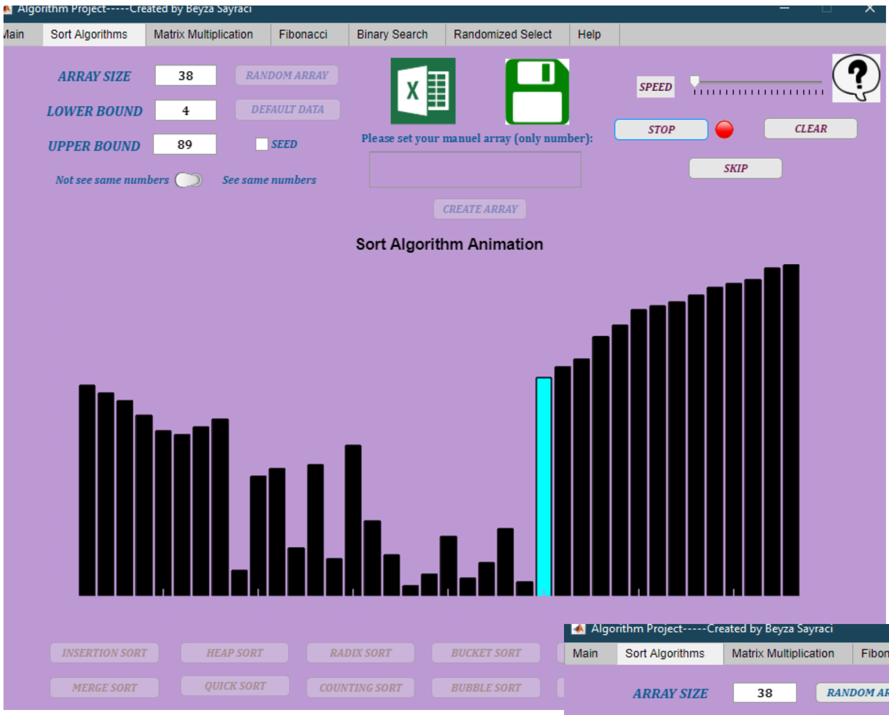


When we click on the **stop button**, our animation in motion stops. In addition, when we run any sort algorithm, all buttons are disabled except skip, stop-continue and clear buttons. It becomes enabled again after the sequence is finished. Example **set (app.COUNTINGSORTButton, 'Enable', 'off'); with disable set (app.COUNTINGSORTButton, 'Enable', 'on'); We do it by enable it**. The function I use for the stop button is; **uiwait** ;

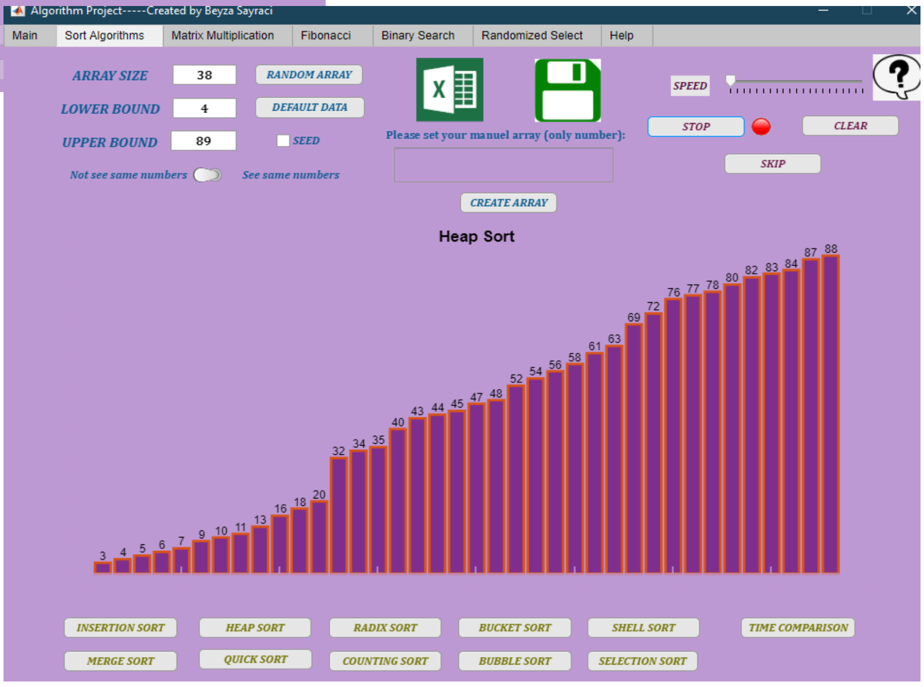
```
uiwait(app.AlgorithmProjectCreatedbyBeyzaSayraciUIFigure);
```

2- Continue Button:

Our stop button turns into a continue button after this event. When we press the same button again, our lamp turns red and the animation continues where it left off. The function I use for the continue button is; **uiresume**;



ARRAY SIZE 38
LOWER BOUND 4
UPPER BOUND 89
SEED
Not see same numbers See same numbers
Please set your manuel array (only number):
SPEED



ARRAY SIZE 38
LOWER BOUND 4
UPPER BOUND 89
SEED
Not see same numbers See same numbers
Please set your manuel array (only number):
SPEED

```
uiresume(app.AlgorithmProjectCreatedbyBeyzaSayraciUIFigure);
```

3- Skip Button:

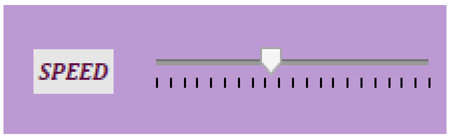
With the skip button banana we can sort our array without waiting for the animation. We do this by clearing the graph with the cla function and reprinting the ordered state of the array on the graph.

```
cla(app.UIAxes2)
app.unsorted_array = sort(app.unsorted_array);
hold(app.UIAxes2,"on")
bar(app.UIAxes2,app.unsorted_array,'FaceColor',"[0.4940 0.
hold(app.UIAxes2,"off")
```

Other features for my bar animation are;

1-Speed

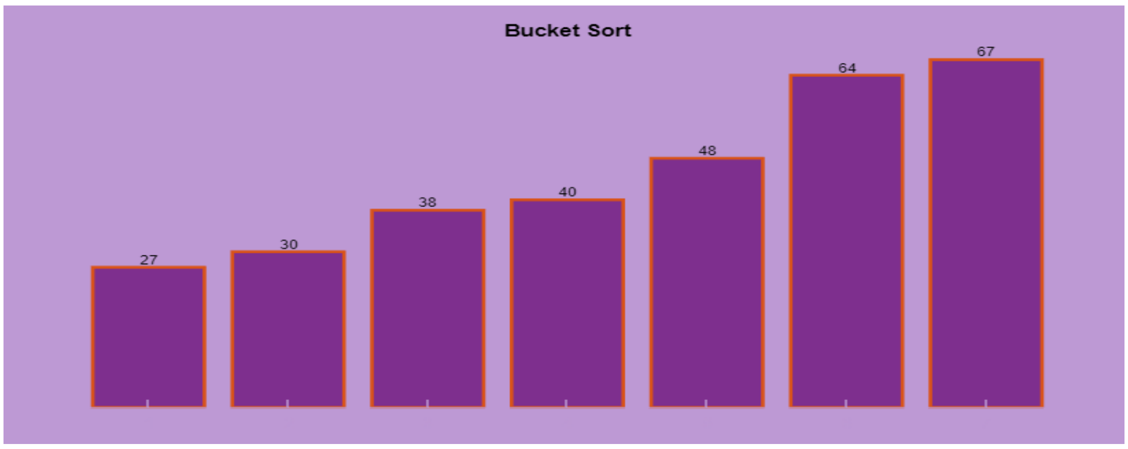
Thanks to Speed, we can adjust the speed of our animation during animation. I used **pause function** for speed.



```
pause((10-floor(app.SPEEDSlider.Value))*0.01);
```

2-Graph Name

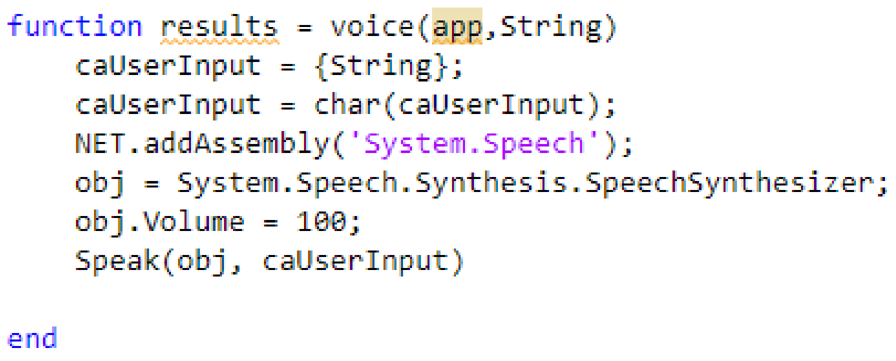
After the animation is finished, the name of the animation is written on the graphic, whichever animation we list.



```
title(app.UIAxes2,'Bucket Sort');
```

3- Sound Effect

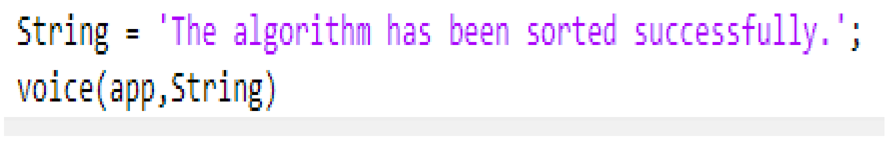
After the animation is finished, you will hear the sound of " **The algorithm has been sorted successfully.**" with the sound effect. For this, I wrote the function in Figure 16. Then I run the function by calling my sort algorithms [Figure 17].



```
function results = voice(app,String)
    caUserInput = {String};
    caUserInput = char(caUserInput);
    NET.addAssembly('System.Speech');
    obj = System.Speech.Synthesis.SpeechSynthesizer;
    obj.Volume = 100;
    Speak(obj, caUserInput)

end
```

Figure 16

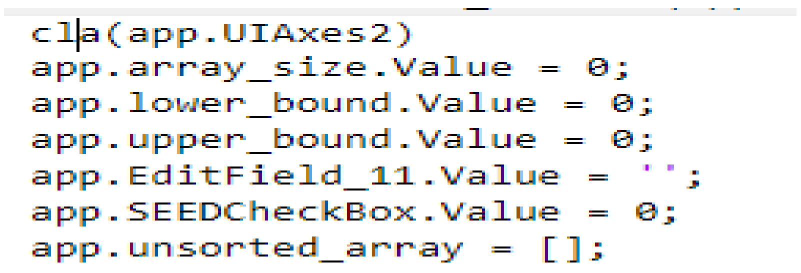


```
String = 'The algorithm has been sorted successfully.';
voice(app,String)
```

Figure 17

4- Clear Button

Thanks to the Clear button, we can clean our interface. The functions I use for this are as seen in the code below.



```
cla(app.UIAxes2)
app.array_size.Value = 0;
app.lower_bound.Value = 0;
app.upper_bound.Value = 0;
app.EditField_11.Value = '';
app.SEEDCheckBox.Value = 0;
app.unsorted_array = [];
```

5- Information Button

When we click on the Information button, a short information box page opens. In this way, we can see how each sort algorithm sorted with videos[Figure 18]. By clicking the Home button, we return to our home page.

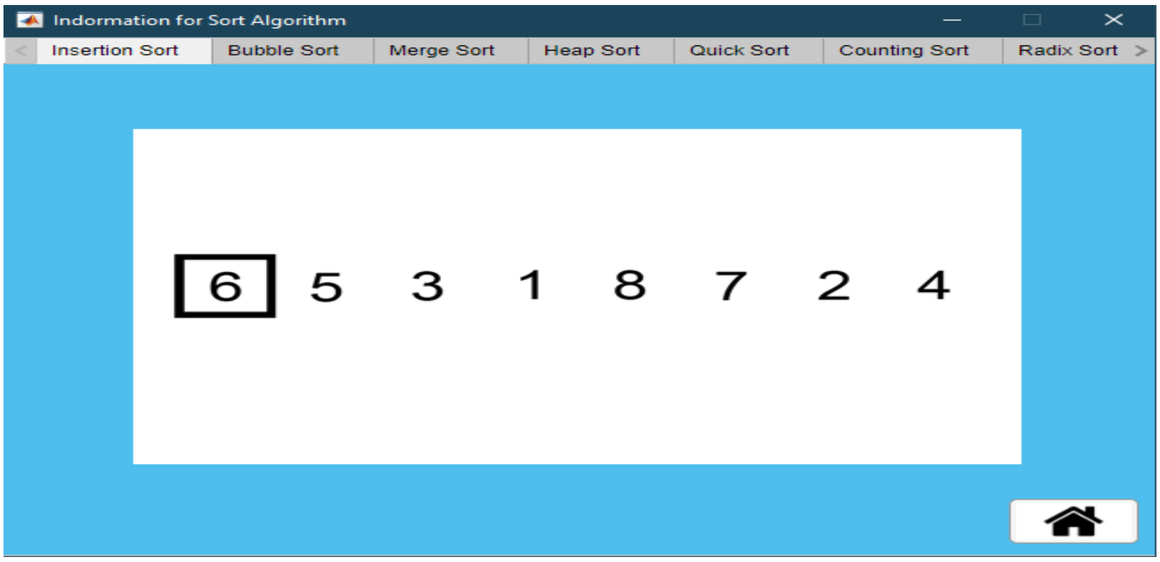
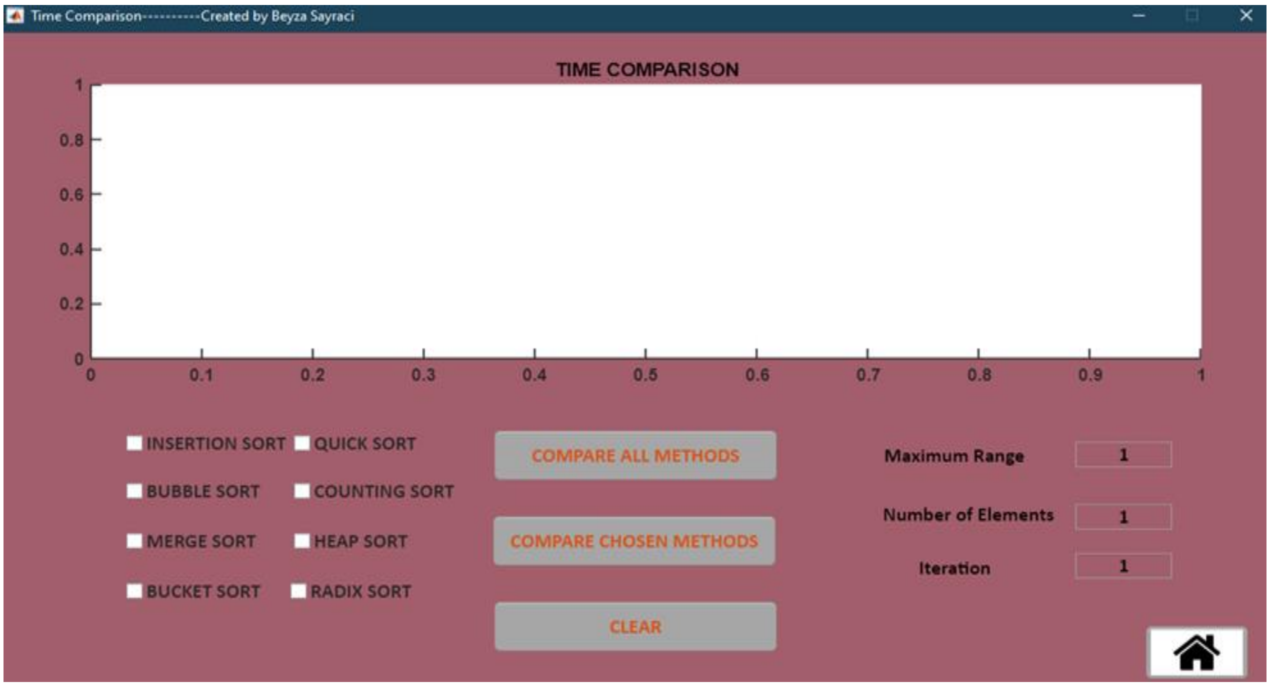


Figure 18

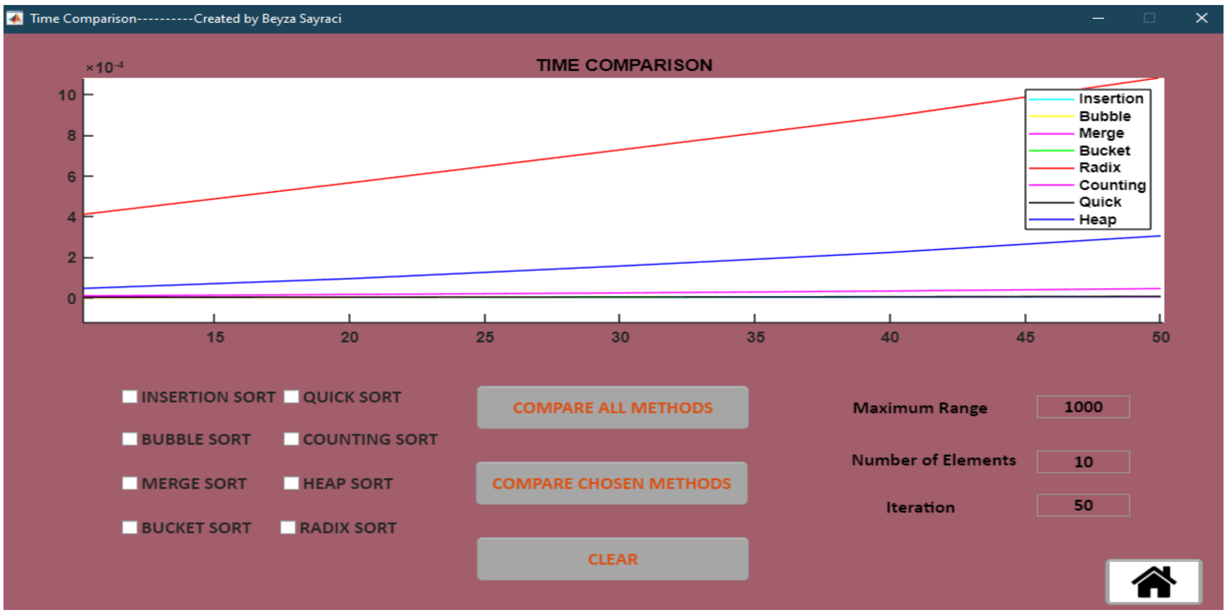
6- Time Comparison Button

When we click on time comparison, a new page opens in front of us.



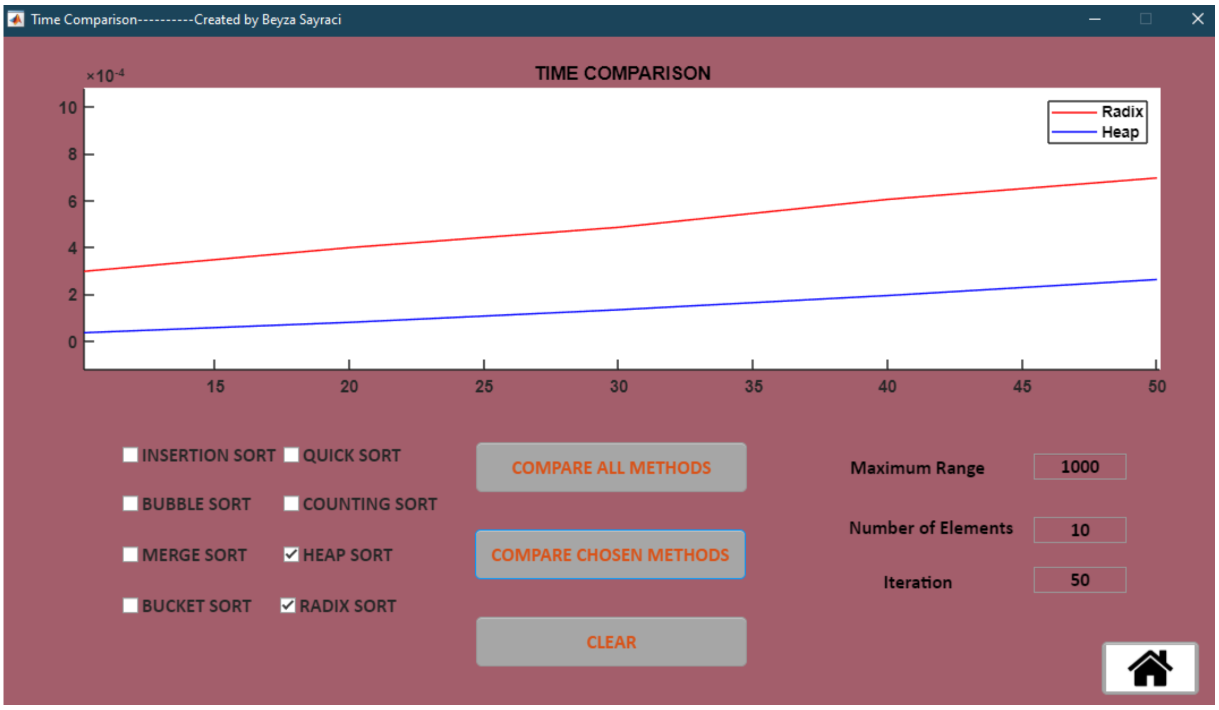
Here the times of the sorting algorithms are compared. After entering the range, size and iteration values, it compares all sort algorithms if we say compare all methods, compares the sort algorithms we choose if we say compare chosen methods.

I clicked compare all methods;



Maximum Range 1000
Number of Elements 10
Iteration 50

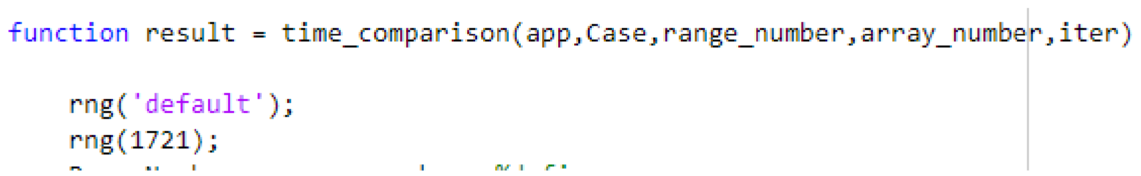
I clicked compare chosen methods;



Maximum Range 1000
Number of Elements 10
Iteration 50

HEAP SORT ✓
RADIX SORT ✓

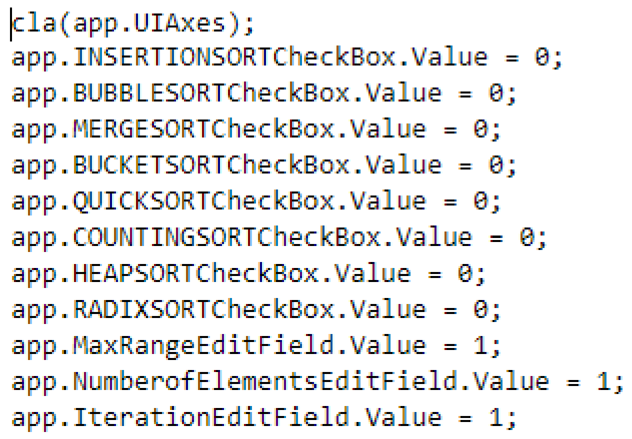
Here, too, I used the rng function so that when the same range, size and iteration values are entered on all computers, the same time comparison can be seen.



```
function result = time_comparison(app,Case,range_number,array_number,iter)

    rng('default');
    rng(1721);
```

Clear button is again useful for cleaning the entire interface;



```
cla(app.UIAxes);
app.INSERTIONSORTCheckBox.Value = 0;
app.BUBBLESORTCheckBox.Value = 0;
app.MERGESORTCheckBox.Value = 0;
app.BUCKETSORTCheckBox.Value = 0;
app.QUICKSORTCheckBox.Value = 0;
app.COUNTINGSORTCheckBox.Value = 0;
app.HEAPSORTCheckBox.Value = 0;
app.RADIXSORTCheckBox.Value = 0;
app.MaxRangeEditField.Value = 1;
app.NumberofElementsEditField.Value = 1;
app.IterationEditField.Value = 1;
```

When I click on the Home button, we go back to the main page.

When we come to our third tab, matrix operations meet us [Figure 19].

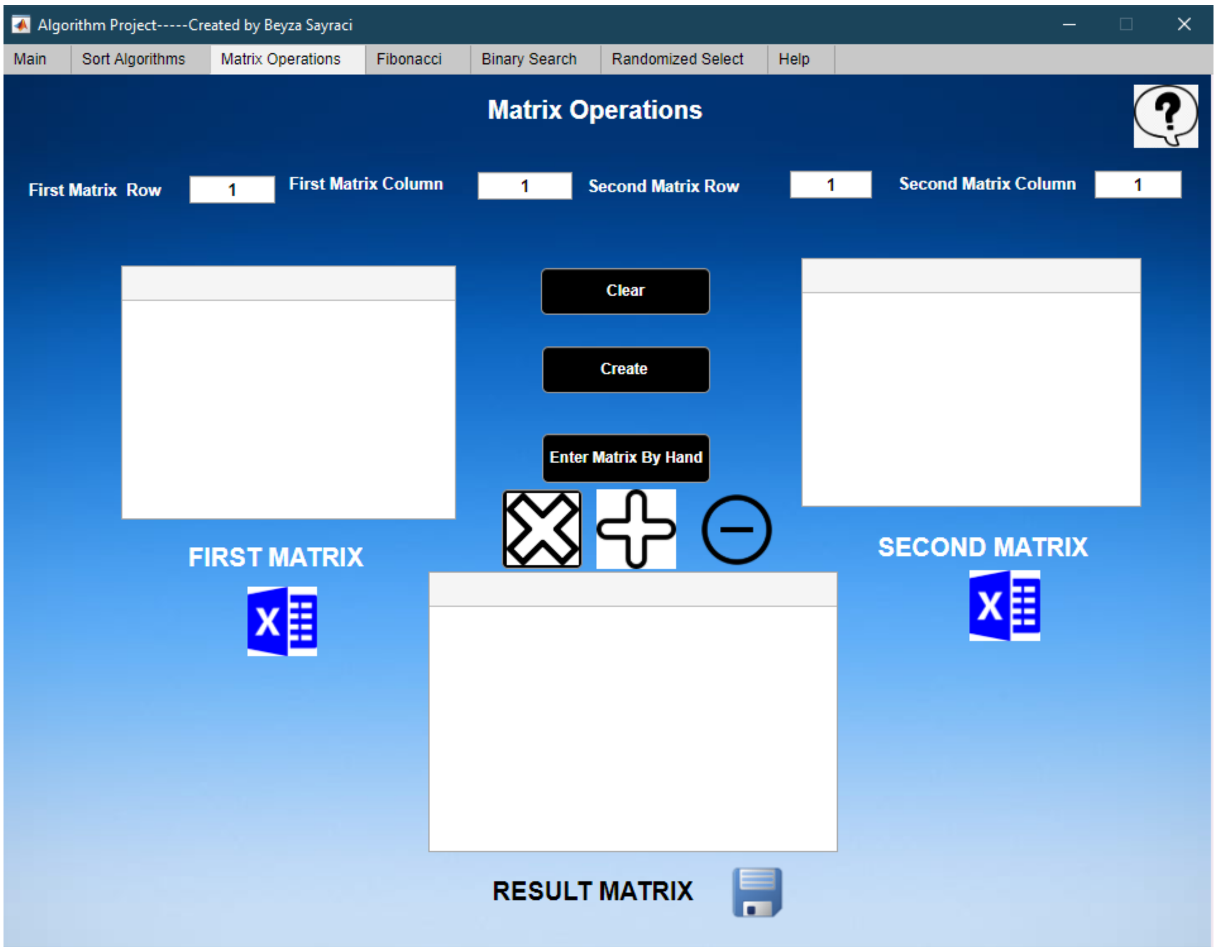
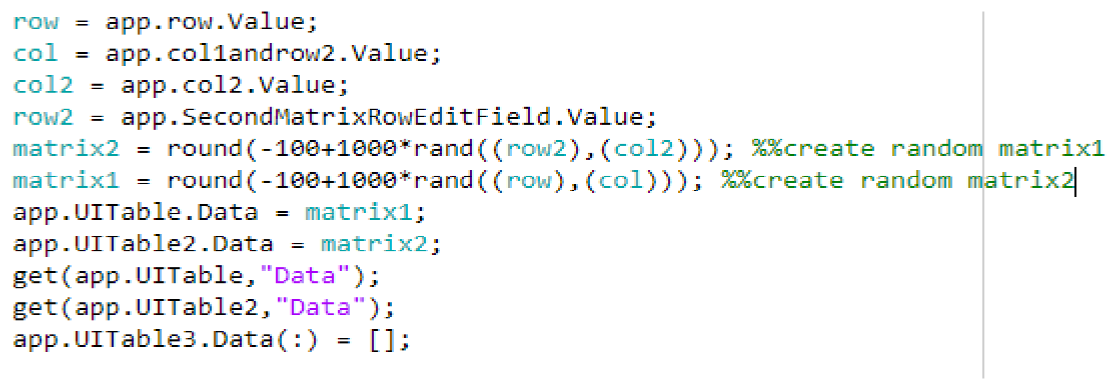


Figure 19

I have two different options to create a matrix, they create a random matrix with the create button and the other is with enter manually matrix. For this, we must first determine the row and cloumn length of our first and second matrices.

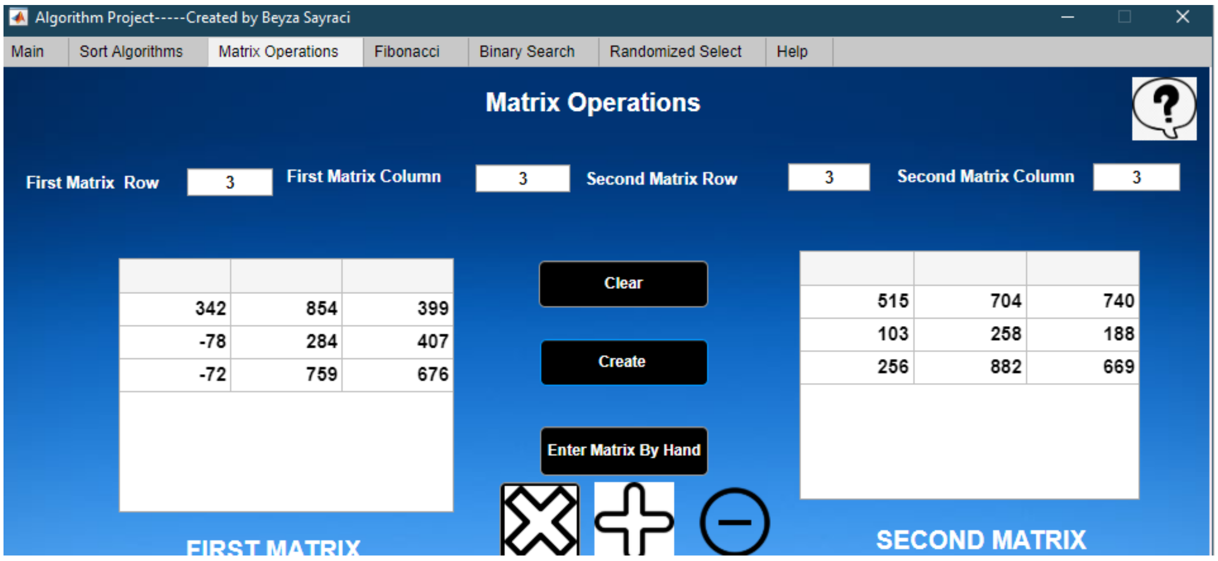
1- Cretae Button



```
row = app.row.Value;
col = app.col1androw2.Value;
col2 = app.col2.Value;
row2 = app.SecondMatrixRowEditField.Value;
matrix2 = round(-100+1000*rand((row2),(col2))); %%create random matrix1
matrix1 = round(-100+1000*rand((row),(col))); %%create random matrix2
app.UITable.Data = matrix1;
app.UITable2.Data = matrix2;
get(app.UITable,"Data");
get(app.UITable2,"Data");
app.UITable3.Data(:) = [];
```

Figure 20

I used the **rand and round functions** to create a random matrix [Figure 20]. I printed the first and second matrices on the tables [Figure 21].



First Matrix Row 3 First Matrix Column 3 Second Matrix Row 3 Second Matrix Column 3

342	854	399
-78	284	407
-72	759	676

515	704	740
103	258	188
256	882	669

Figure 21

2- Enter Matrix By Hand Button

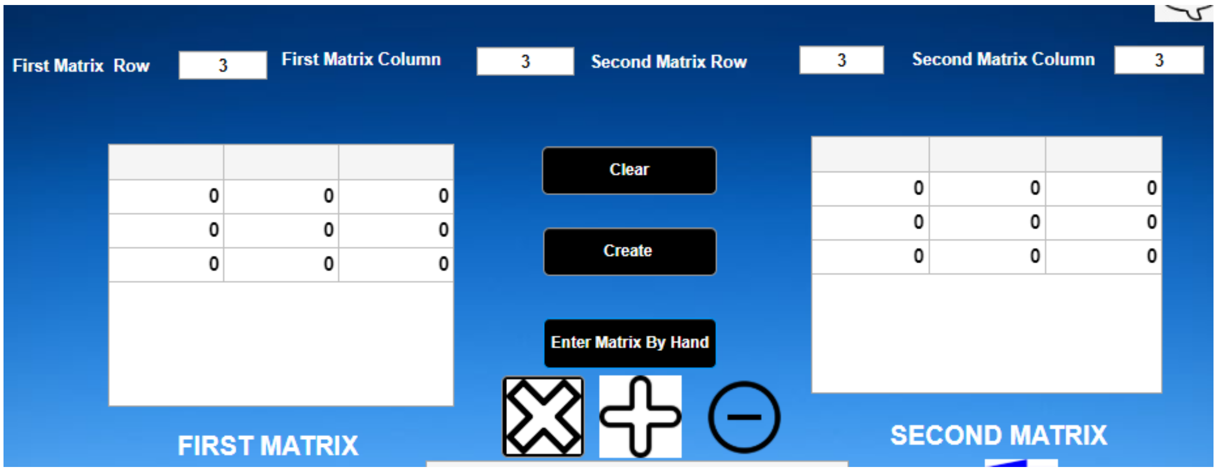
Here, I created an empty matrix according to the length of the row and column with the **zeros function** [Figure 22]. I've printed these blank matrices into the tables. I used **get function** [Figure 23]. In this way, we can create the matrix we want.

```
row = app.row.Value;
col = app.col1androw2.Value;
col2 = app.col2.Value;
matrix1 = zeros(row,col);
matrix2 = zeros(col,col2);
app.UITable.Data = matrix1;
app.UITable2.Data = matrix2;
get(app.UITable,"Data");
get(app.UITable2,"Data");
app.UITable3.Data(:) = [];
```

Figure 22

3- Get Excel

We can call the matrices we saved in the form of xlsx file to our interface.



0	0	0
0	0	0
0	0	0

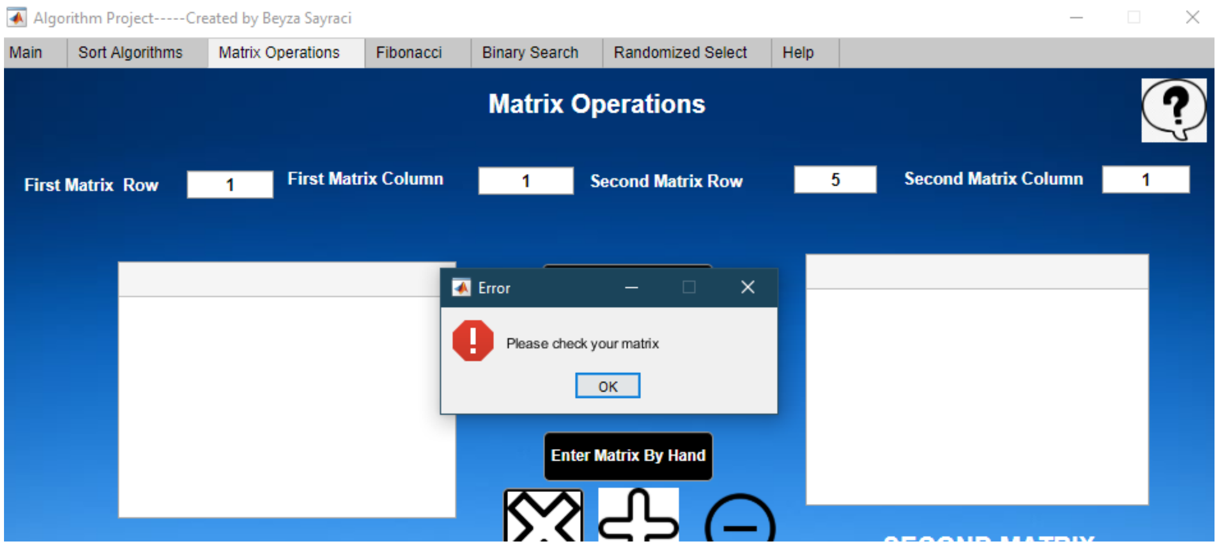
0	0	0
0	0	0
0	0	0

Figure 23

I do three different matrix operations in this matrix operations page. Subtraction, addition and multiplication

Let's examine the multiplication first,

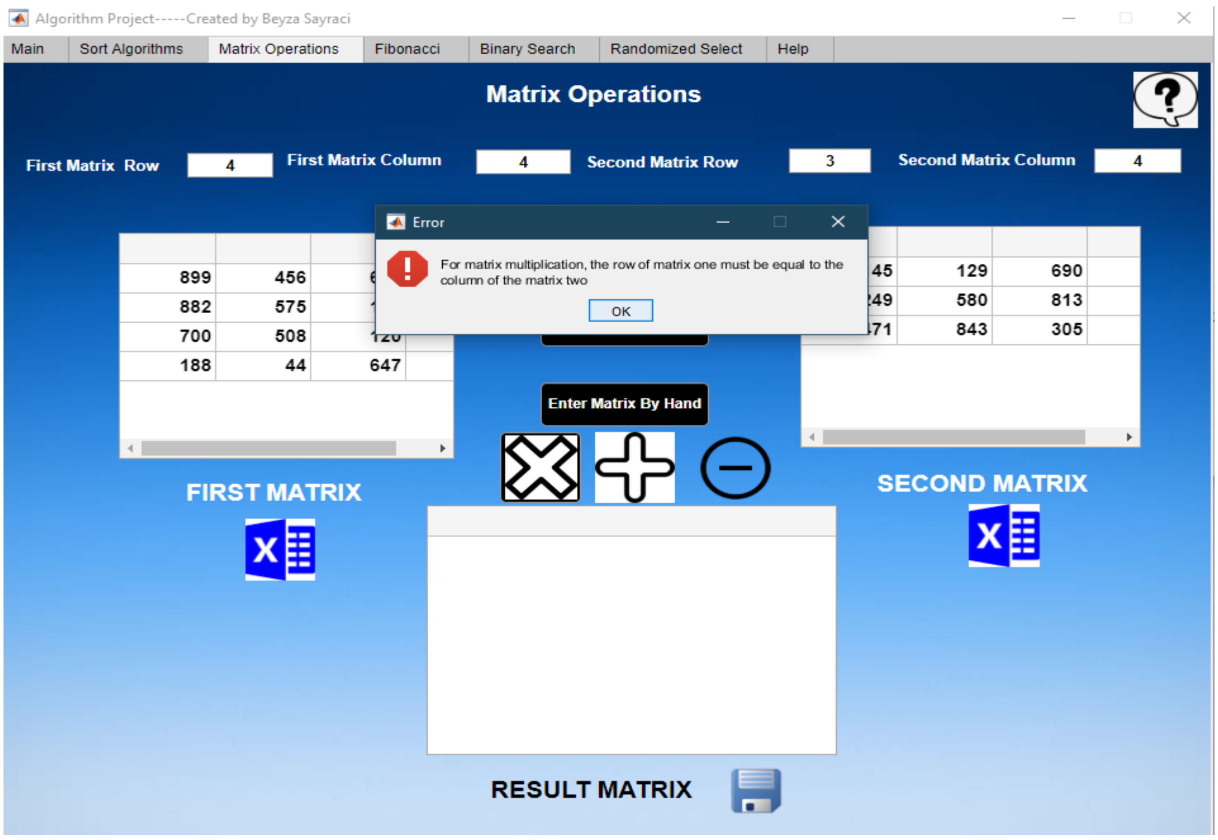
We click on our multiplication button to multiply the matrix that we created manually or randomly. If we click the multiplication button without creating any matrix, we will see an error message [Figure 24].



Please check your matrix

Figure 24

If the column length of the first matrix and row length of the second matrix are not equal to each other, we will encounter an error message again [Figure 25]. Because by the matrix multiplication rule, they must be equal.



First Matrix Row 4 First Matrix Column 4 Second Matrix Row 3 Second Matrix Column 4

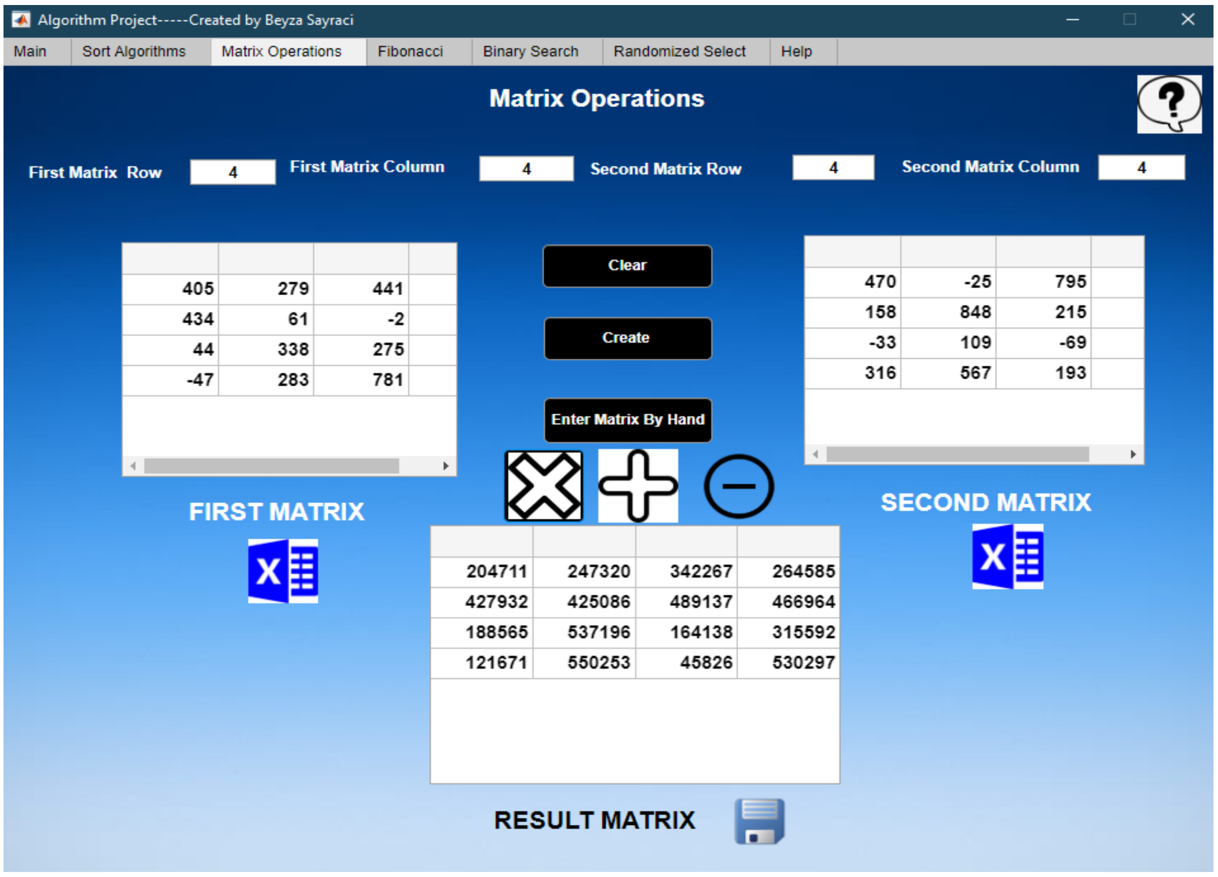
For matrix multiplication, the row of matrix one must be equal to the column of the matrix two

899	456	6
882	575	1
700	508	120
188	44	647

45	129	690
249	580	813
71	843	305

Figure 25

If we got everything right, we do matrix multiplication [Figure 26].



First Matrix Row 4 First Matrix Column 4 Second Matrix Row 4 Second Matrix Column 4

405	279	441
434	61	-2
44	338	275
-47	283	781

470	-25	795
158	848	215
-33	109	-69
316	567	193

204711	247320	342267	264585
427932	425086	489137	466964
188565	537196	164138	315592
121671	550253	45826	530297

Figure 26

If we want, we can save our result in an excel file.

When the necessary rules for matrix sum and matrix subtraction are met, an error message is displayed. If the row and column lengths according to the required rules are entered, our operations are done.

When you click on the information box in the upper right corner, short information about matrix operations is reached.

When we come to our fourth tab, fibonacci meet us [Figure 27];

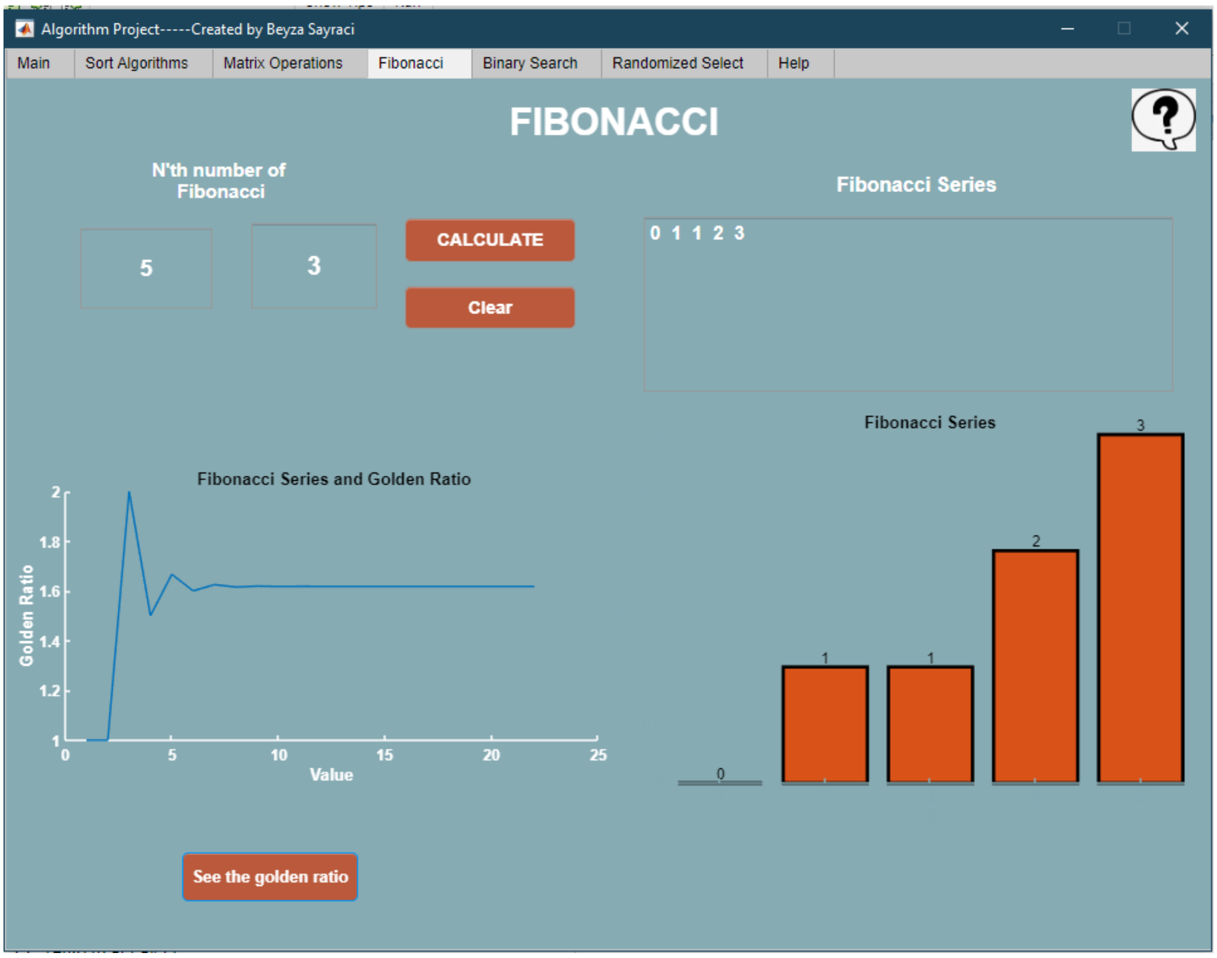
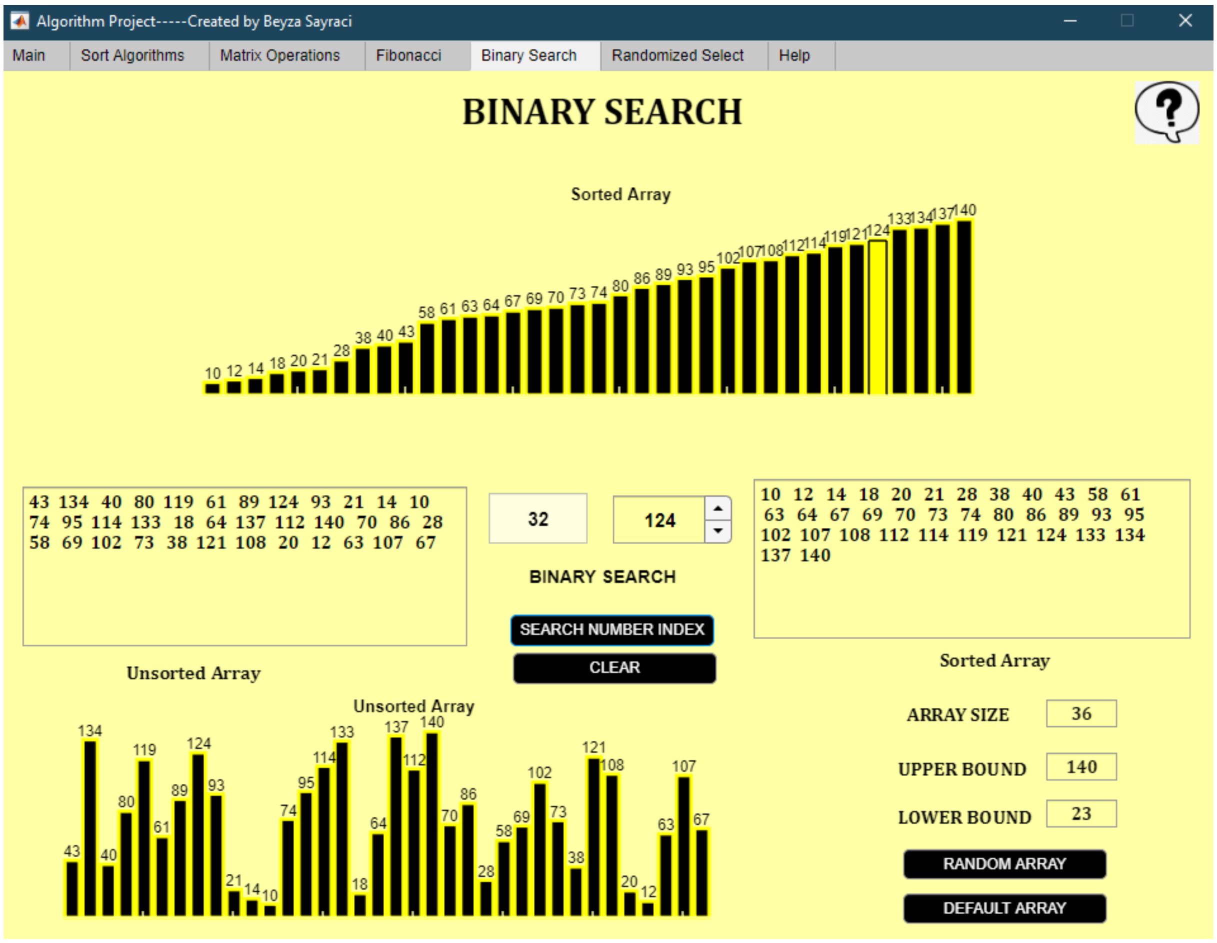


Figure 27

When we enter the index we want to see in the Fibonacci series, we can see both the number and the list. We also observe the list in the form of bars. It's available on the golden ratio chart on the same page. The Clear button clears the whole screen, the information box gives information about the fibonacci series.

When we come to our fifth tab, Binary search meet us [Figure 28];



43 134 40 80 119 61 89 124 93 21 14 10
74 95 114 133 18 64 137 112 140 70 86 28
58 69 102 73 38 121 108 20 12 63 107 67

32

124

10 12 14 18 20 21 28 38 40 43 58 61
63 64 67 69 70 73 74 80 86 89 93 95
102 107 108 112 114 119 121 124 133 134
137 140

ARRAY SIZE 36

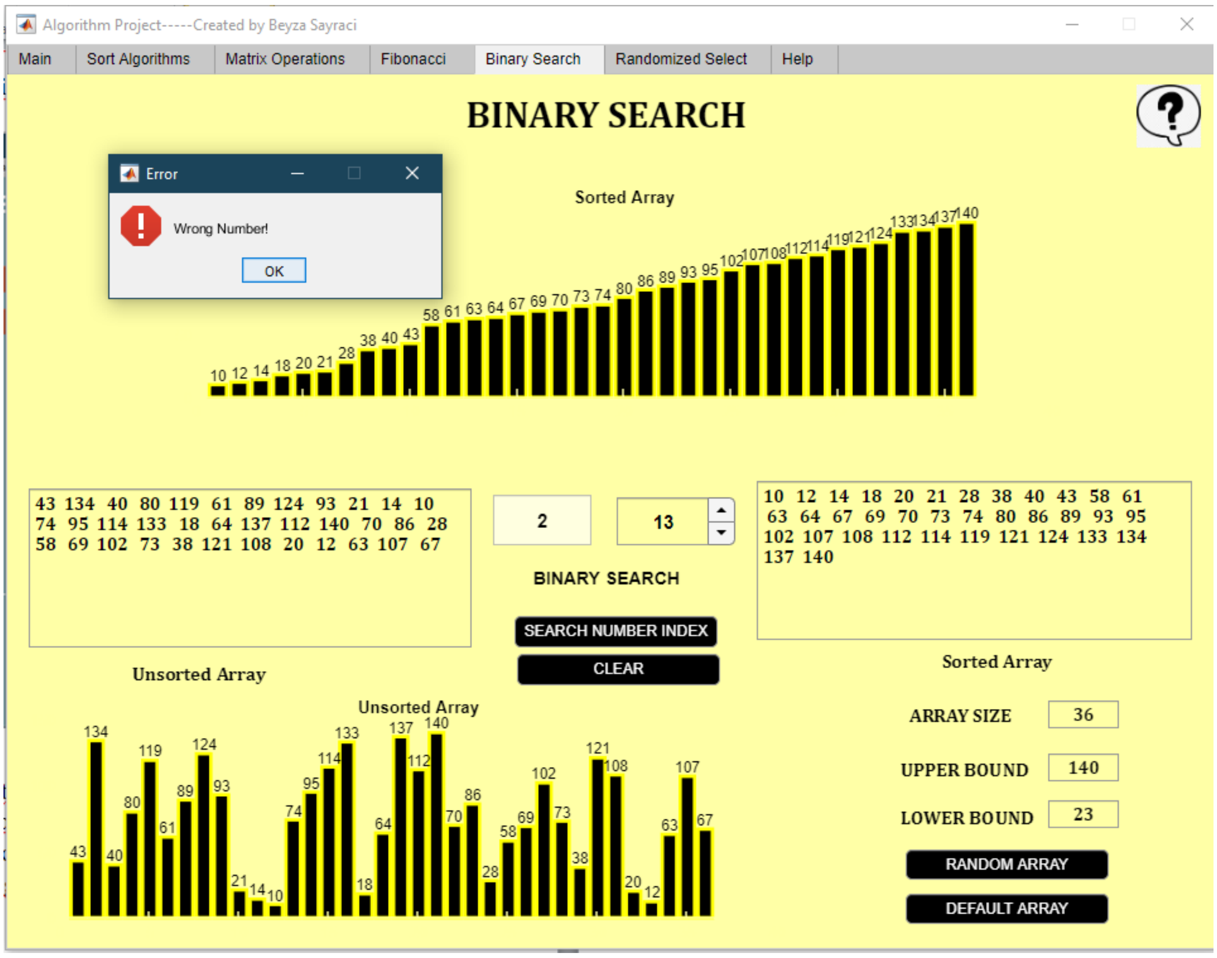
UPPER BOUND 140

LOWER BOUND 23

Figure 28

In our binary search screen, we can create two types of arrays. Random array and default array. Error messages in the sorting algorithm are also present here. Also, in the arrays we have created, the same numbers never appear. There are completely different numbers coming up. I did this with the functions I use in the sorting algorithm. We observe our unsorted and sorted arrays both as a list and a bar. Also, when we enter the number we are looking for, the sorted array bar also paints the number. If the number we are looking for is not available in the list, the color disappears [Figure 29].

Our clear button clears the whole screen, we can access information about binary search with our information box [Figure 30]



43 134 40 80 119 61 89 124 93 21 14 10
74 95 114 133 18 64 137 112 140 70 86 28
58 69 102 73 38 121 108 20 12 63 107 67

2   13

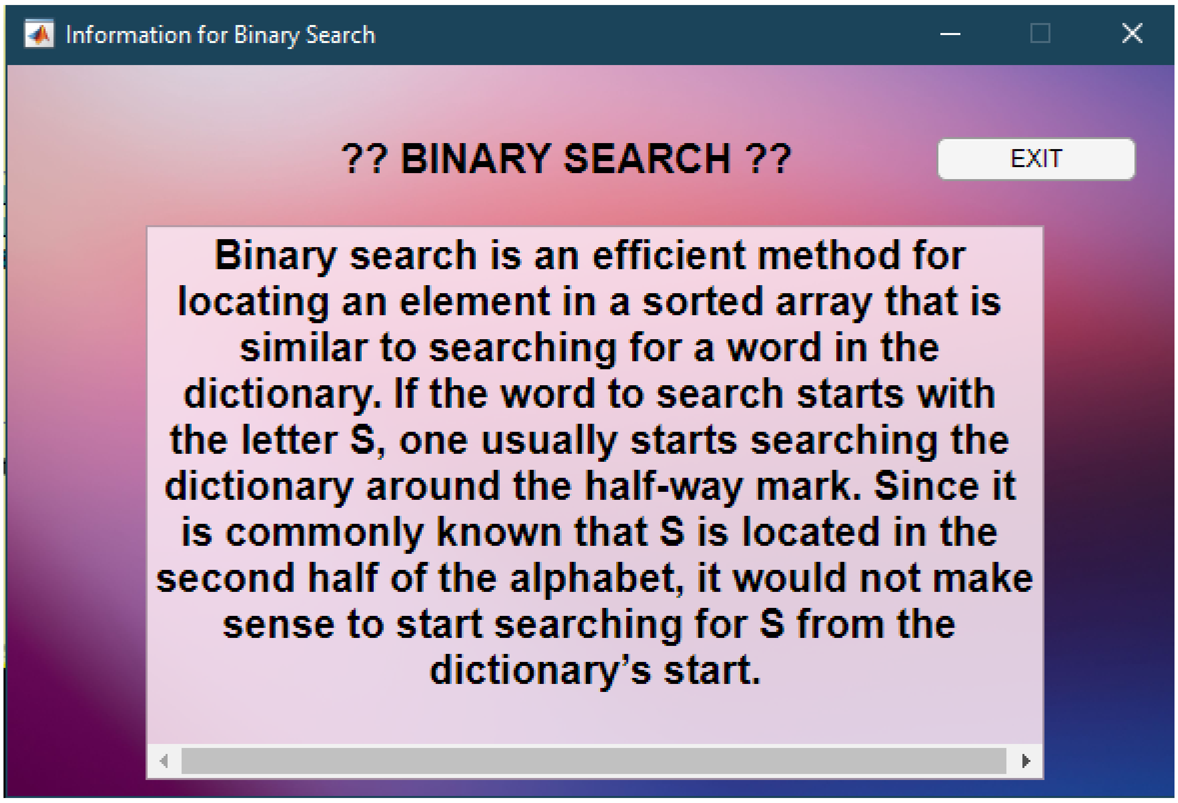
10 12 14 18 20 21 28 38 40 43 58 61
63 64 67 69 70 73 74 80 86 89 93 95
102 107 108 112 114 119 121 124 133 134
137 140

ARRAY SIZE 36

UPPER BOUND 140

LOWER BOUND 23

Figure 29

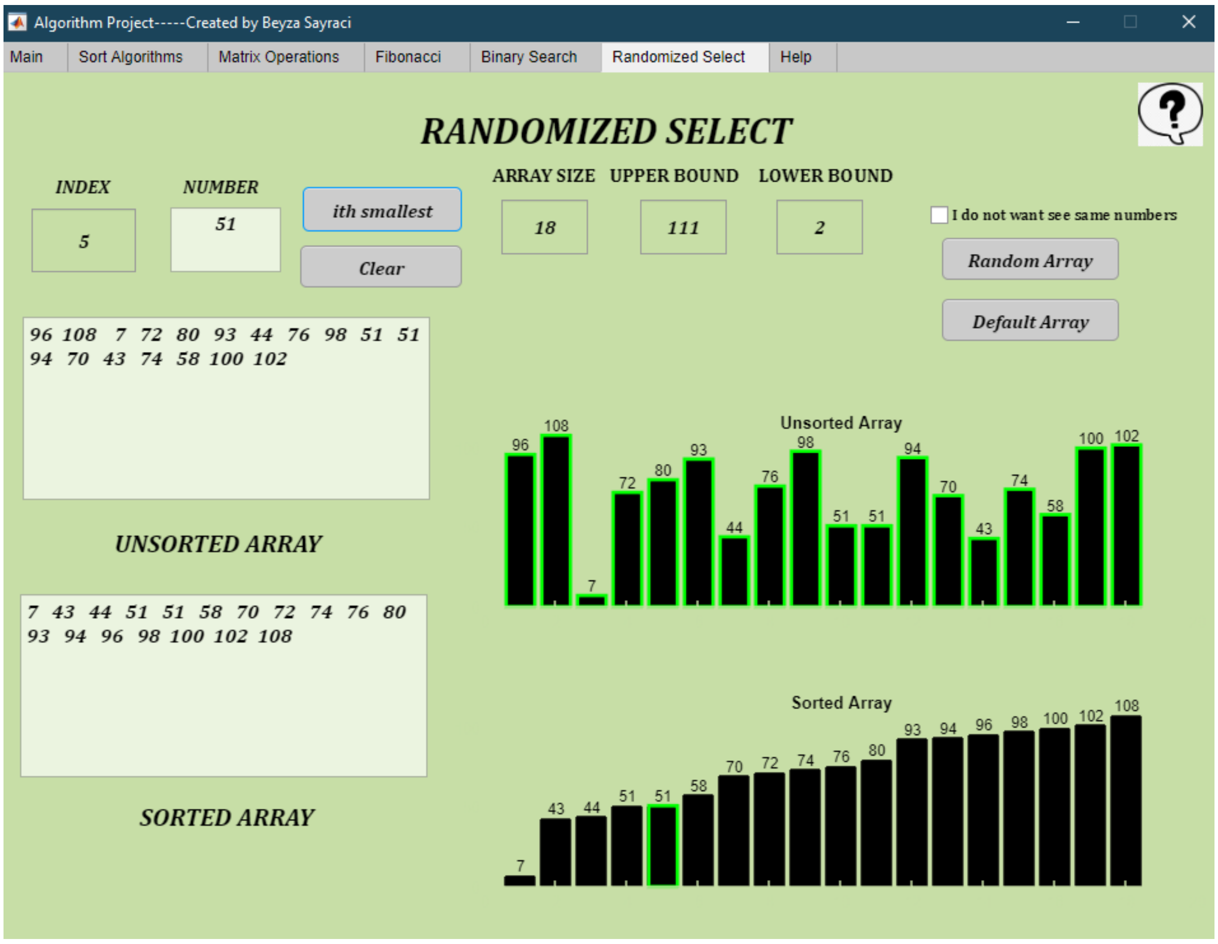


?? BINARY SEARCH ??

Binary search is an efficient method for locating an element in a sorted array that is similar to searching for a word in the dictionary. If the word to search starts with the letter S, one usually starts searching the dictionary around the half-way mark. Since it is commonly known that S is located in the second half of the alphabet, it would not make sense to start searching for S from the dictionary’s start.

Figure 30

When we come to our sixth tab, Randomized Select meet us [Figure 31]



INDEX 5

NUMBER 51

ARRAY SIZE 18

UPPER BOUND 111

LOWER BOUND 2

```
96 108  7  72  80  93  44  76  98  51  51
94  70  43  74  58 100 102
```

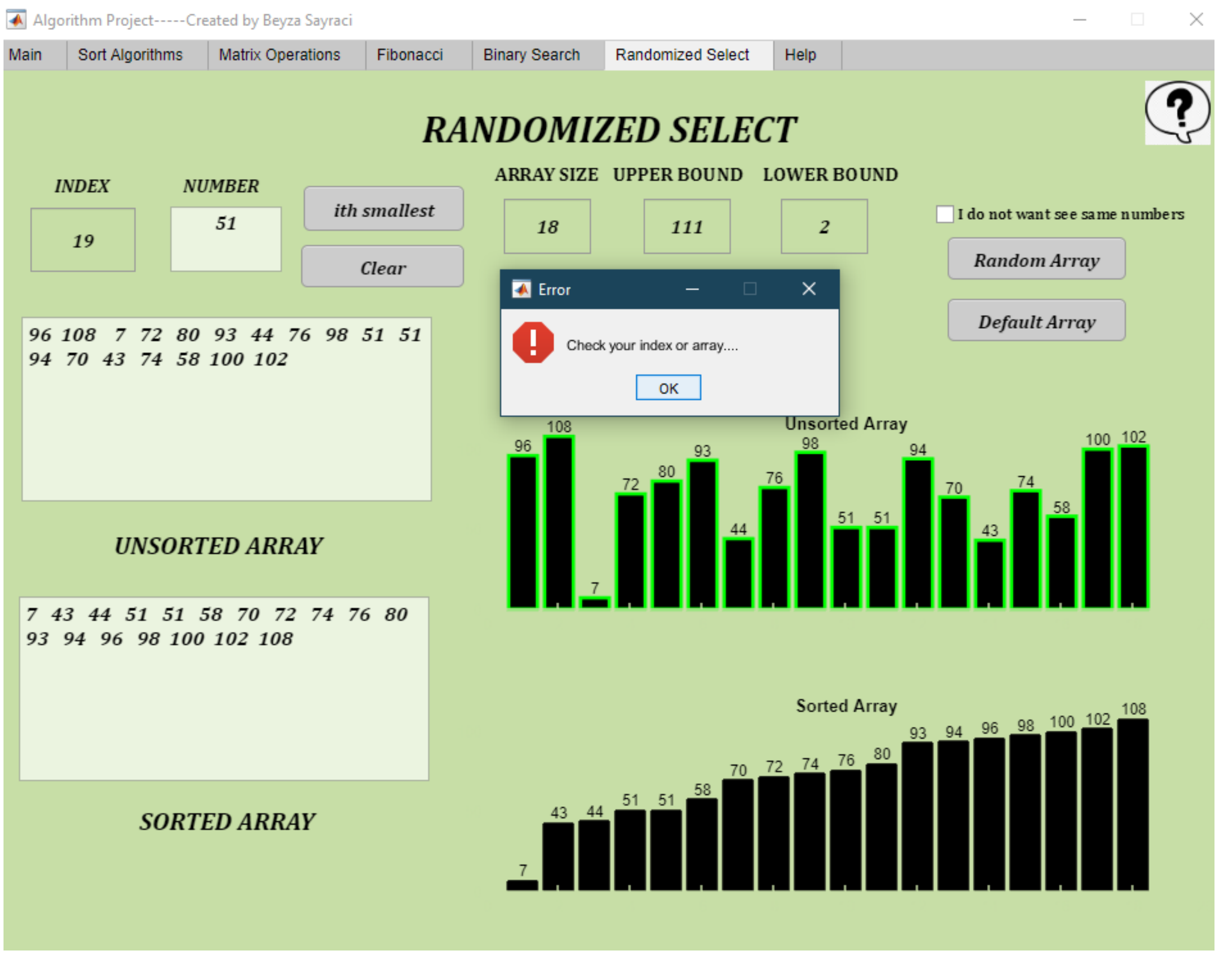
UNSORTED ARRAY

```
7  43  44  51  51  58  70  72  74  76  80
93  94  96  98 100 102 108
```

SORTED ARRAY

Figure 31

We can create two different arrays in randomized select. Random array and default array. Error messages in the sorting algorithm are also present here. If we click on the I do not see same numbers button, there are no similar numbers in the array we created. I did this with the functions I use in the sorting algorithm. We observe our unsorted and sorted arrays both as a list and a bar. When we enter the index we are looking for, the number in that index is painted in the sorted array bar. If the index we are looking for is not available, an error message is given [Figure 32]. Our clear button clears the whole screen, we can access information about binary search with our information box.



INDEX: 19
NUMBER: 51
ARRAY SIZE: 18
UPPER BOUND: 111
LOWER BOUND: 2

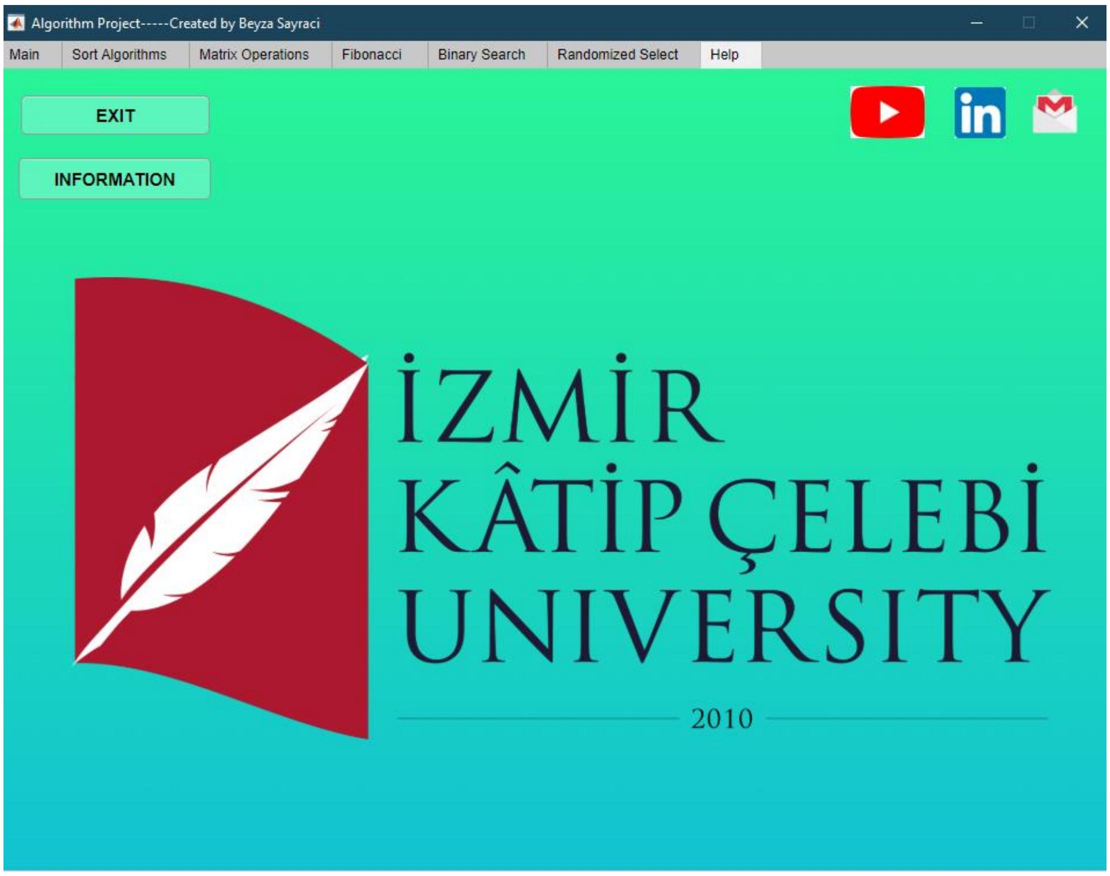
Check your index or array....

96 108 7 72 80 93 44 76 98 51 51 94 70 43 74 58 100 102

7 43 44 51 51 58 70 72 74 76 80 93 94 96 98 100 102 108

Figure 32

When we come to our seven tab, help meet us [Figure 33]



Information section is available here [Figure 34]. You are directed to the pages I use on the information page. You can send me your feedback via the Gmail section [Figure 35]. With the Linkedin button, you can reach my linkedin page with the youtube button on my youtube page and our school page with the ikçü logo.

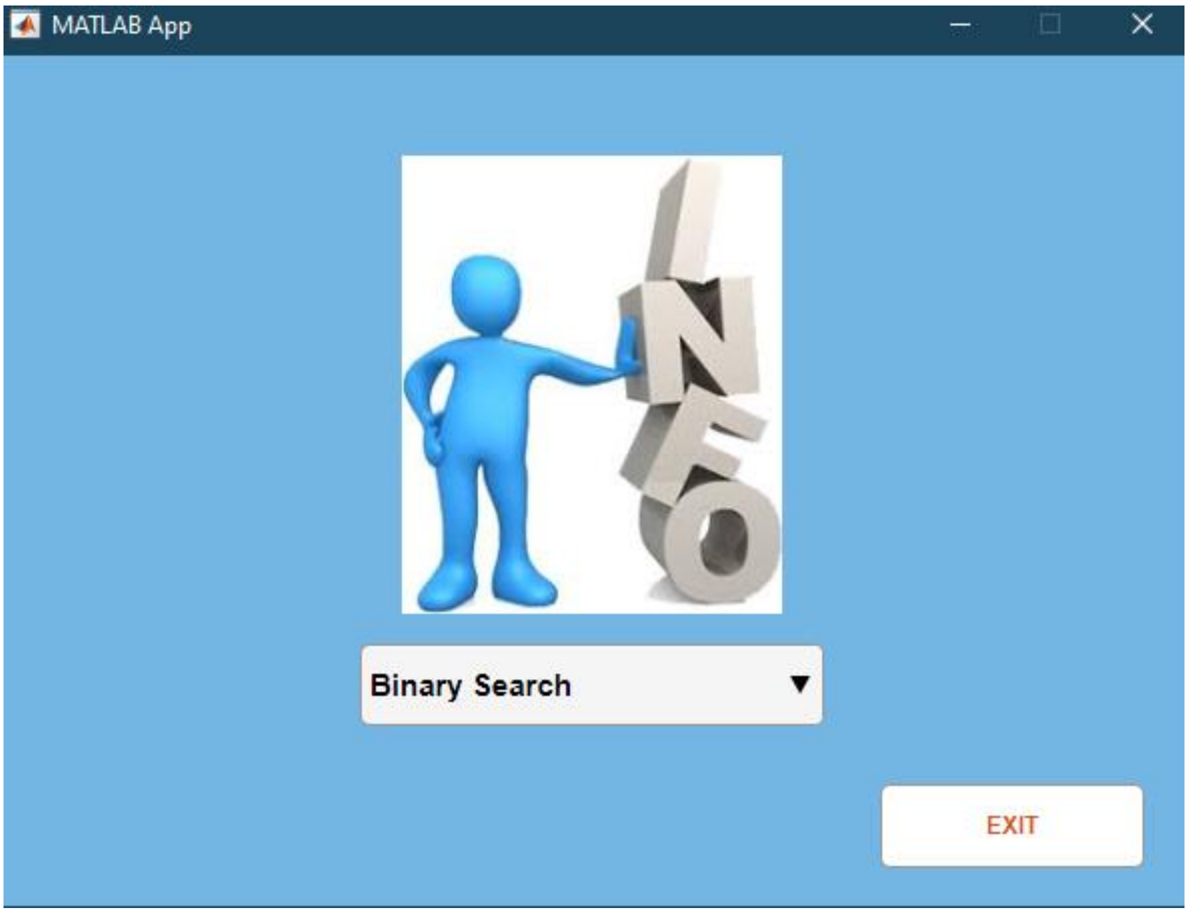


Figure 34

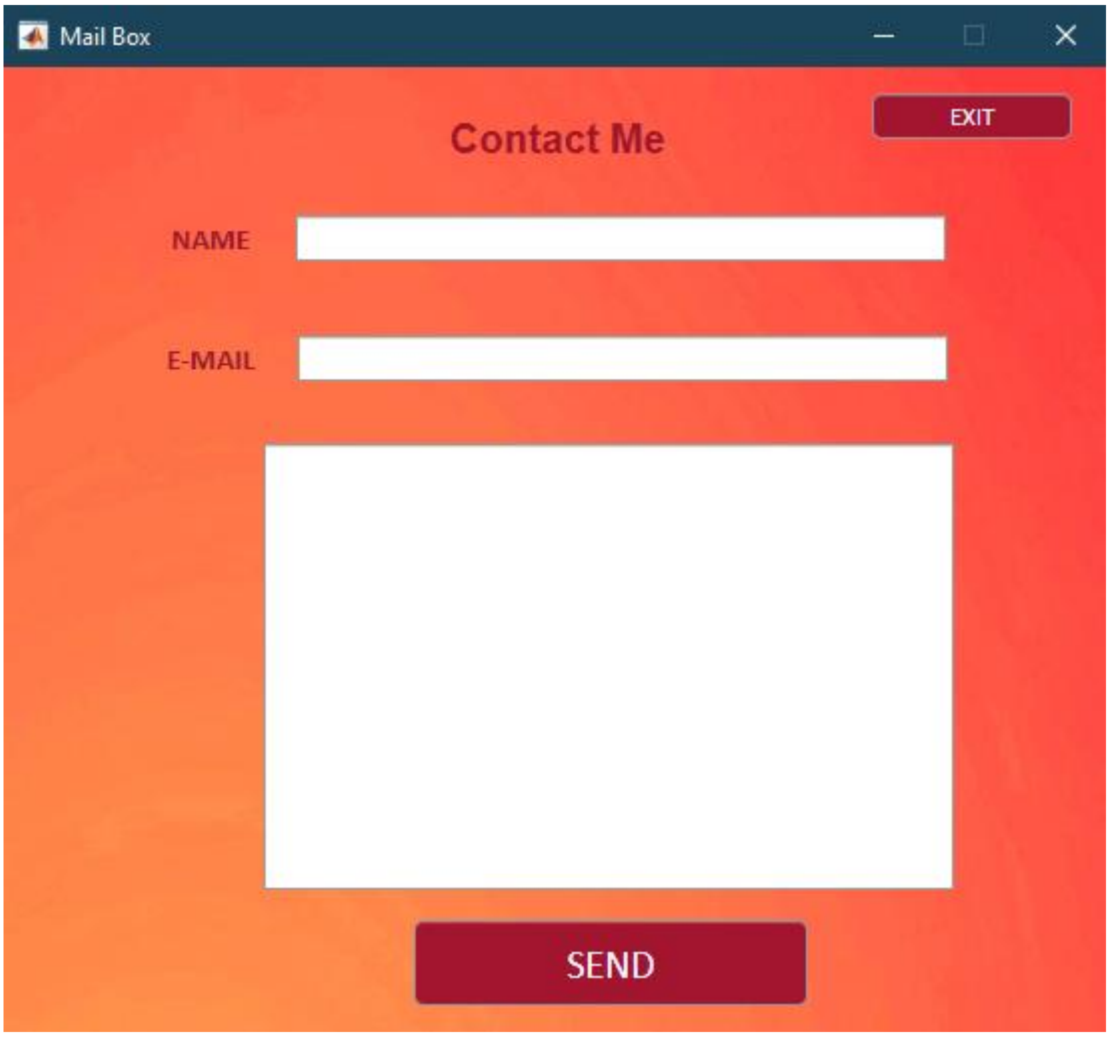


Figure 35

<u>Additional features:</u>

1- I made gradient color backgrounds.

2- Resize screen

3- I destroyed the graphic lines and graphic display

4- I added gifs

5- My interface cannot be trolled.