NEURAL NETWORK

PROJECT 2: HOPFIELD ASSOCIATIVE MEMORY

MENTOR: PROFESSOR MAZEK KOZLOWSKI

STUDENT: BUI TUAN ANH, ID:309051

Contents

1. Hopfield Associative Memory Idea

A Hopfield network is a form of recurrent artificial neural network popularized by John Hopfield in 1982, but described earlier by Little in 1974. Hopfield networks serve as content-addressable ("associative") memory with binary threshold nodes. They are guaranteed to converge to a local minimum and, therefore, may converge to a false pattern (wrong local minimum) rather than the stored pattern (expected local minimum). Hopfield networks also provide a model for understanding human memory.

1.1. Training Phase

Associative memory: elements are addressed by the content of the input rather than a physical address

- The memory should respond with the library (training) vector in case this vector is shown in the input
- The memory should respond with the correct version of a library vector in case the "noisy version" (withing some limits of the an=mount of noise added) of this vector is presented in the input.

Learning rule: In this project, I use Hebb rule for learning. Hebb's postulates:

- In human brains, some psychological concepts may be represented by simultaneous activation of some (a group of) neurons.
- Functional groups of neurons are being formed in the learning process by strengthening connections (weights) of all simultaneously activated neurons.
- Memorized (learned) concept can be recalled if sufficient number (not necessarily all) of neurons representing this conception is simultaneously activated – the so-called context addressing

The Hebbian theory was introduced by Donald Hebb in 1949, in order to explain "associative learning", in which simultaneous activation of neuron cells leads to pronounced increases in synaptic strength between those cells. The description of Hebb's rule is below:

Set X composed of M bipolar vectors $X^i = [x_i^1, \dots, x_i^N], i = 1, \dots, M$. A memory is composed of N neurons $neu_1, \dots, neu_N$. Weight matrix T is defined by:

$$t_{ij} = \begin{cases} 0 \; for \; i = j \\ \frac{1}{N}\sum_{s=1}^{M} x_i^s x_i^s \;\; for \; i \neq j \end{cases} \quad (1.1)$$

for i,j = 1,…..,N

1.2. Recognition (Test) Phase

This project uses sign function which is described below:

$$g(x) = \begin{cases} 1 \; if \; x > 0 \\ -1 \; if \; x <= 0 \end{cases} \quad (1.2)$$

For instance, we have result vector $v = (v_1, \dots, v_N)$, v is calculated by:

$$v_i = g(u_i)$$

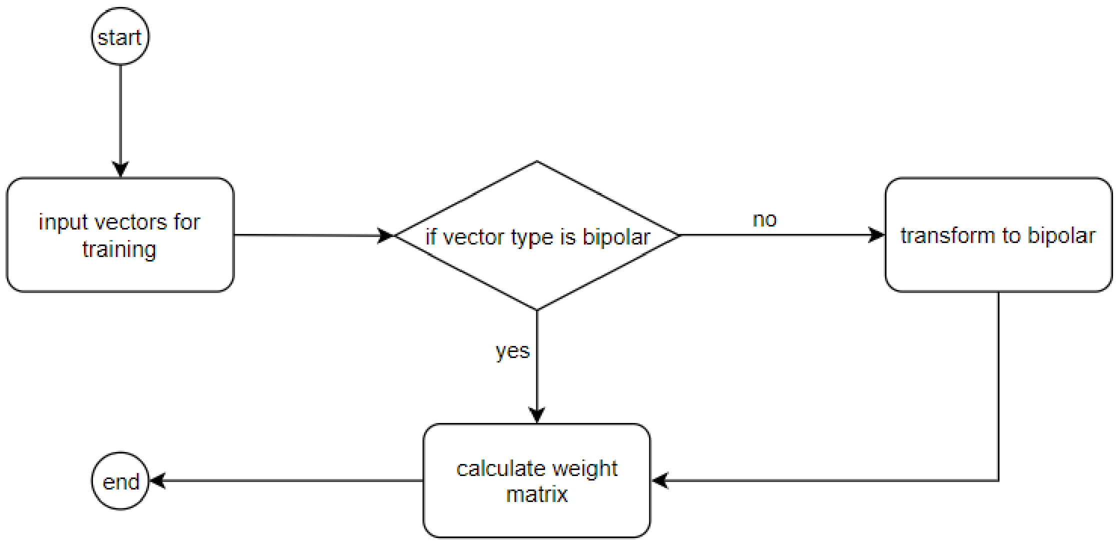
$$u_i = \sum_{j=1}^{N} t_{ij} z_j \,, i = 1, \dots, N \quad (1.3)$$

There are 2 possible scenarios:

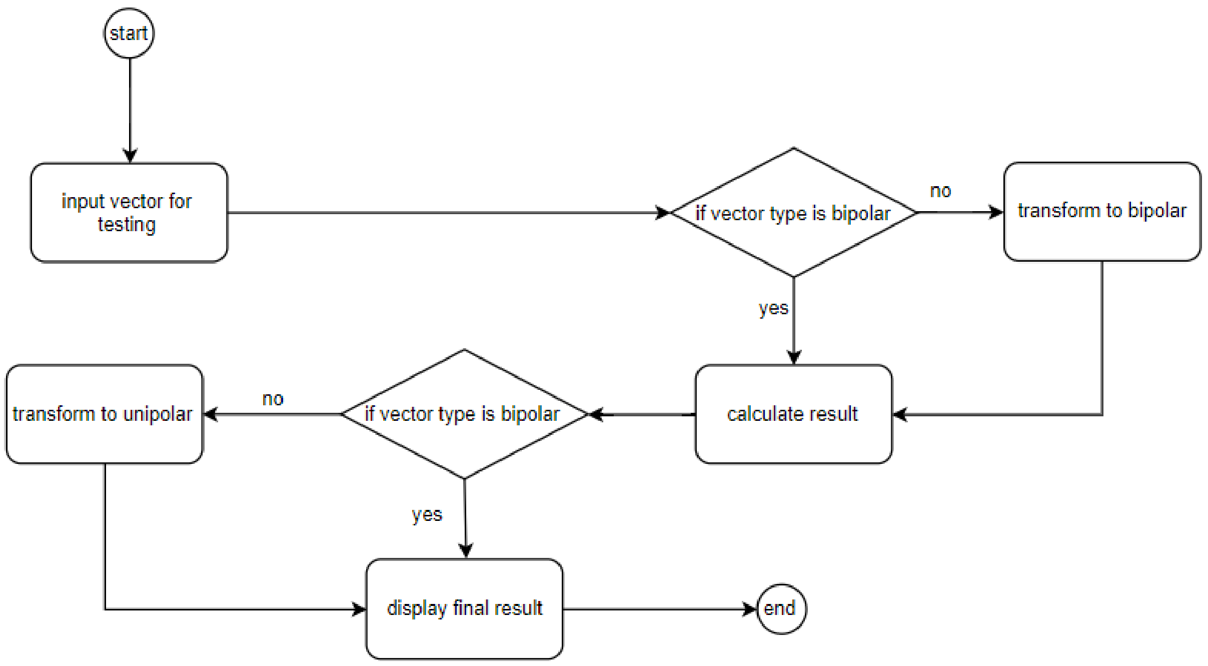
- Either the output equals the input
- Or a cycle of length 2: A-B-A-B….

In the program, user can use both bipolar (1, -1) and unipolar (1, 0) vectors for training and testing. While training, if the inputs vectors are bipolar, the upper formula will be applied without any transformation. If the inputs vectors are unipolar, the upper formula will be applied after transformation of inputs vectors into bipolar (transform 0 to -1). The flow will be like:

For training:

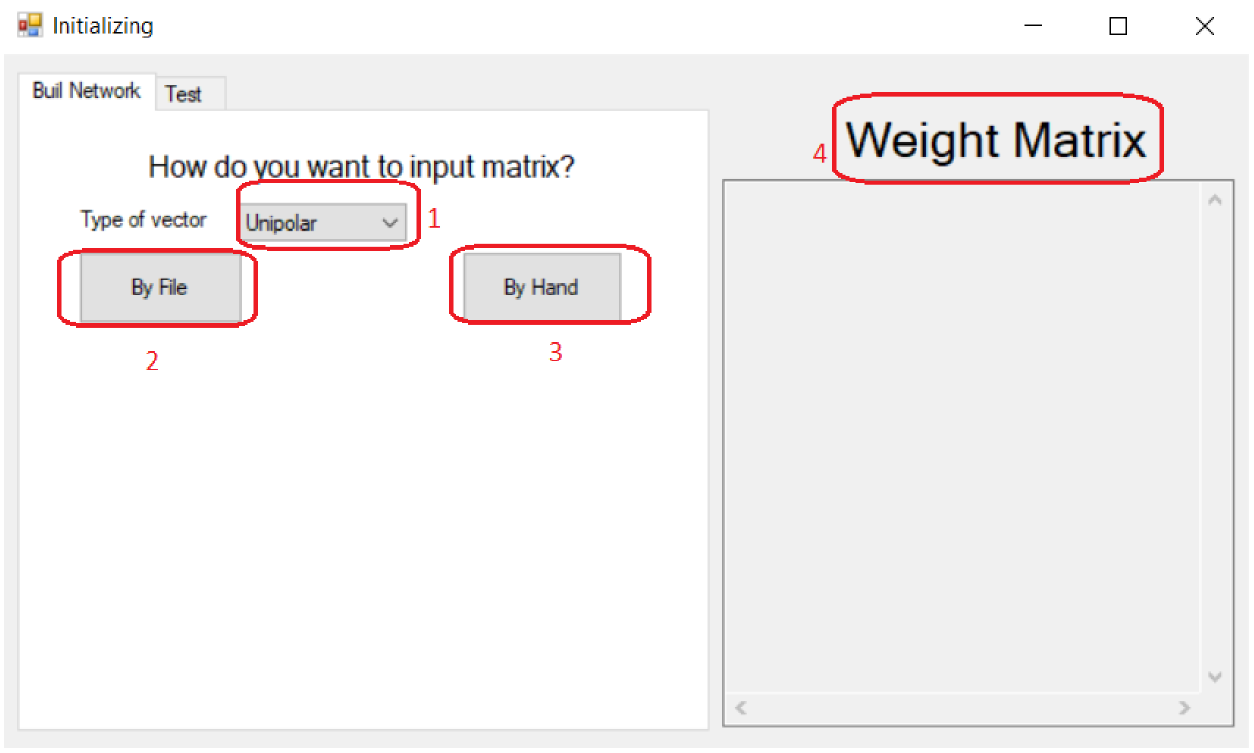


For testing:



2. User Interface

This is GUI when user run the program:

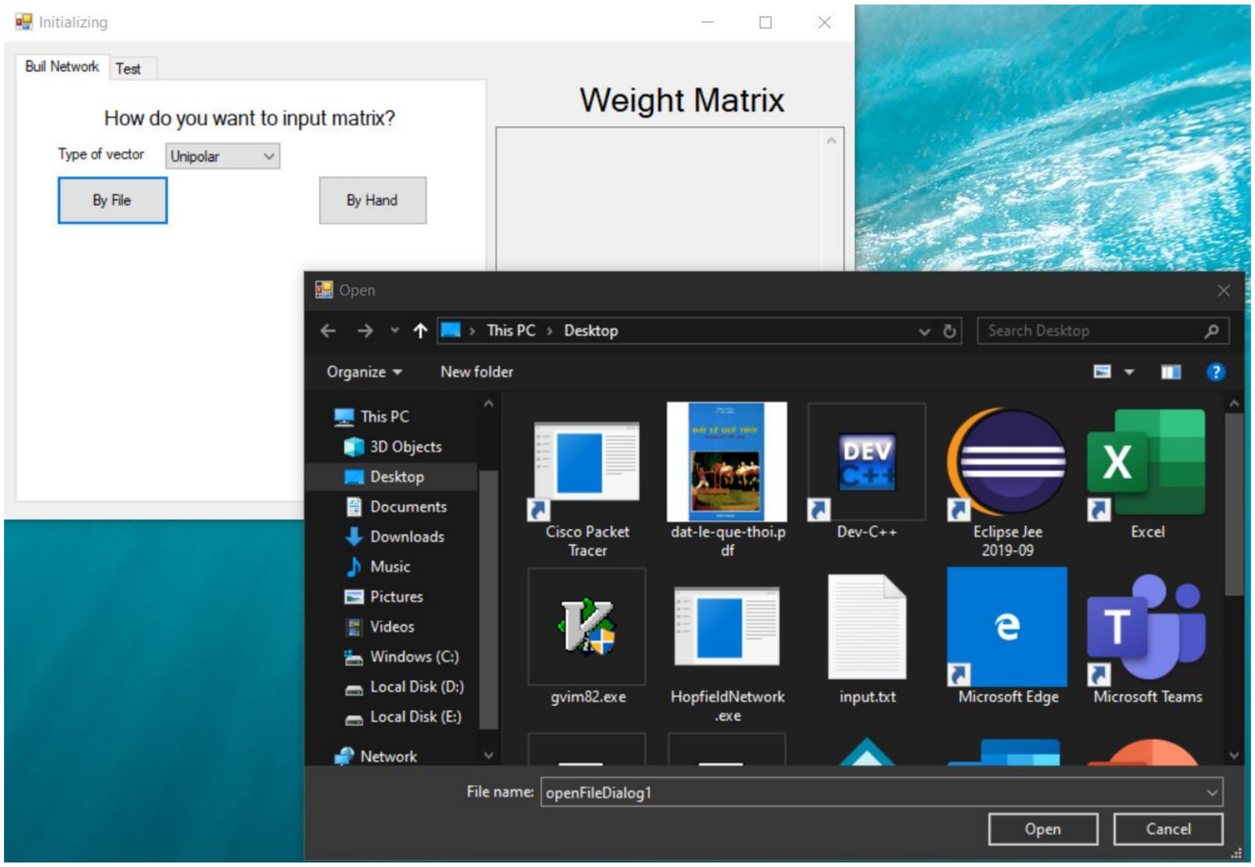


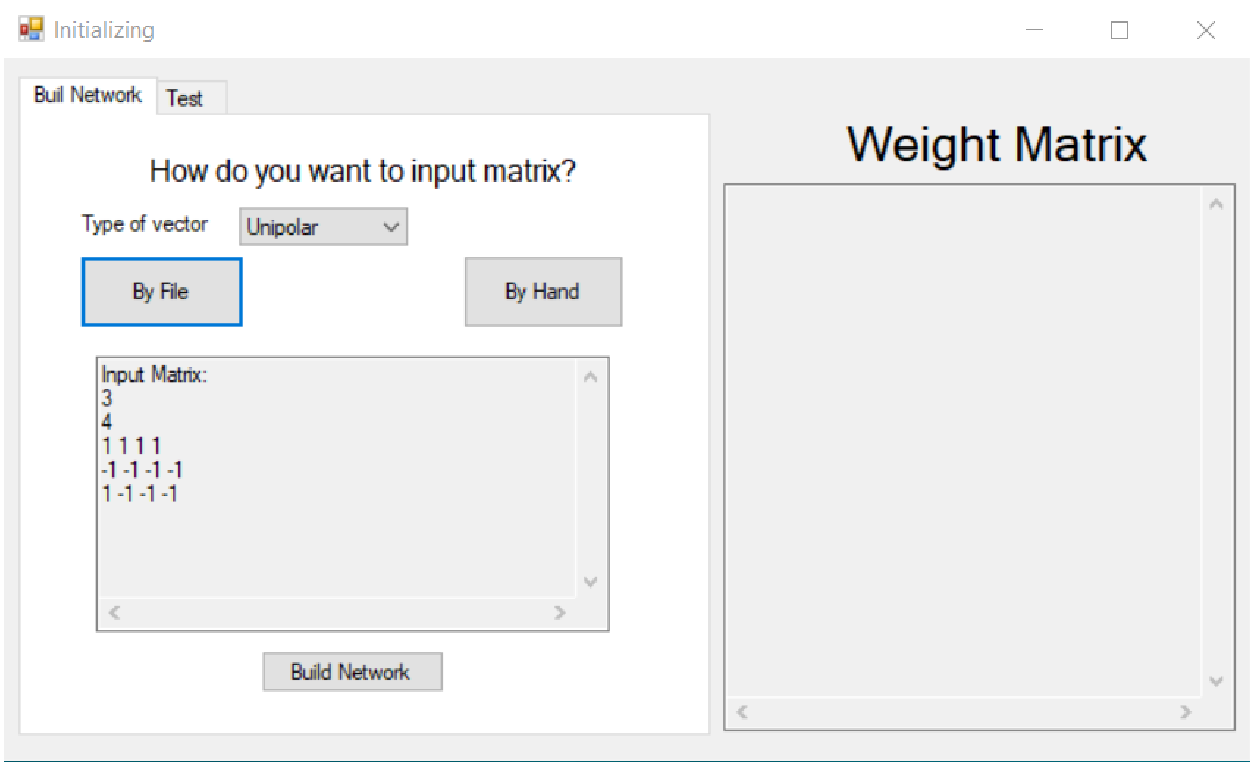
There are 2 tabs, 1 for building network and 1 for testing.

2.1. Build Network Tab

In the building tab, combo box 1 allows user to choose type of vector. There are 2 buttons 2 and 3 for choosing how to input user's matrix: "By File" and "By Hand". Text box 4 for display weight matrix result.

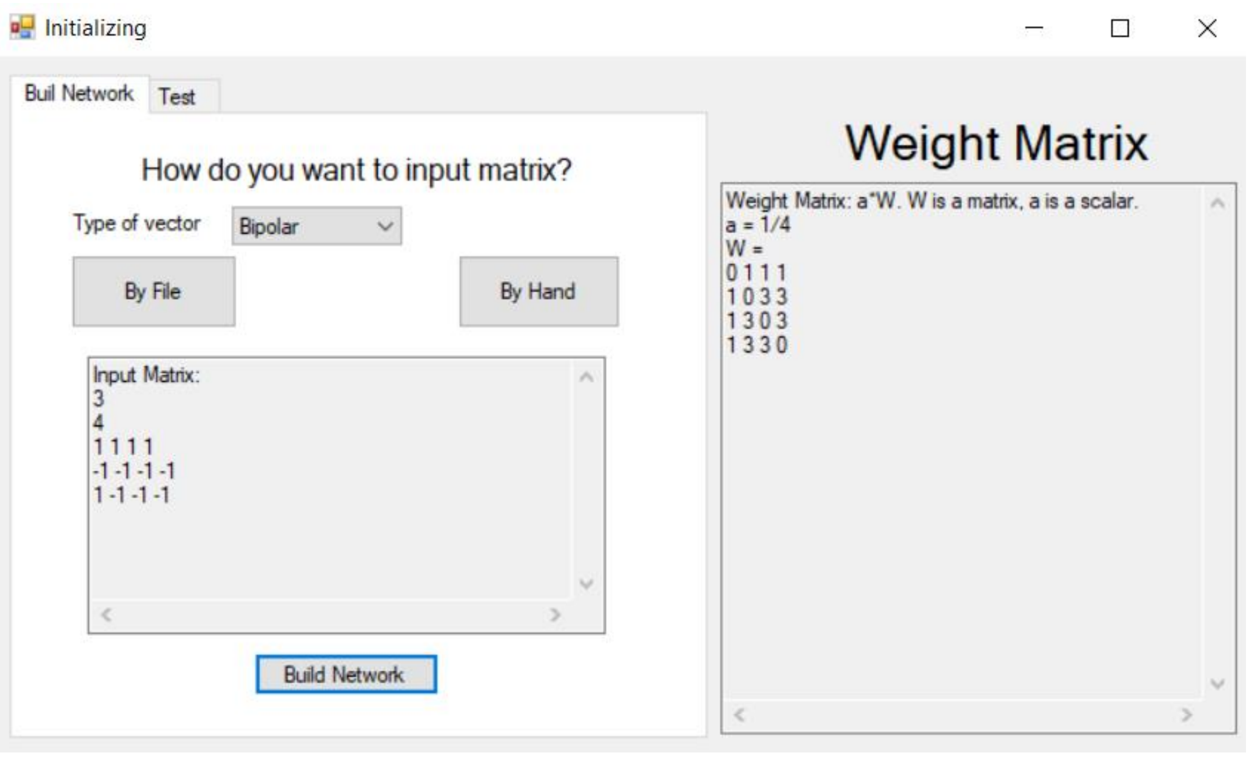
If user click "By File" button, a dialog will be showed for choosing file. After choosing a file, the file's detail will be displayed with a button "Build Network" for calculate weight matrix.





```
Input Matrix:
3
4
1 1 1 1
-1 -1 -1 -1
1 -1 -1 -1
```

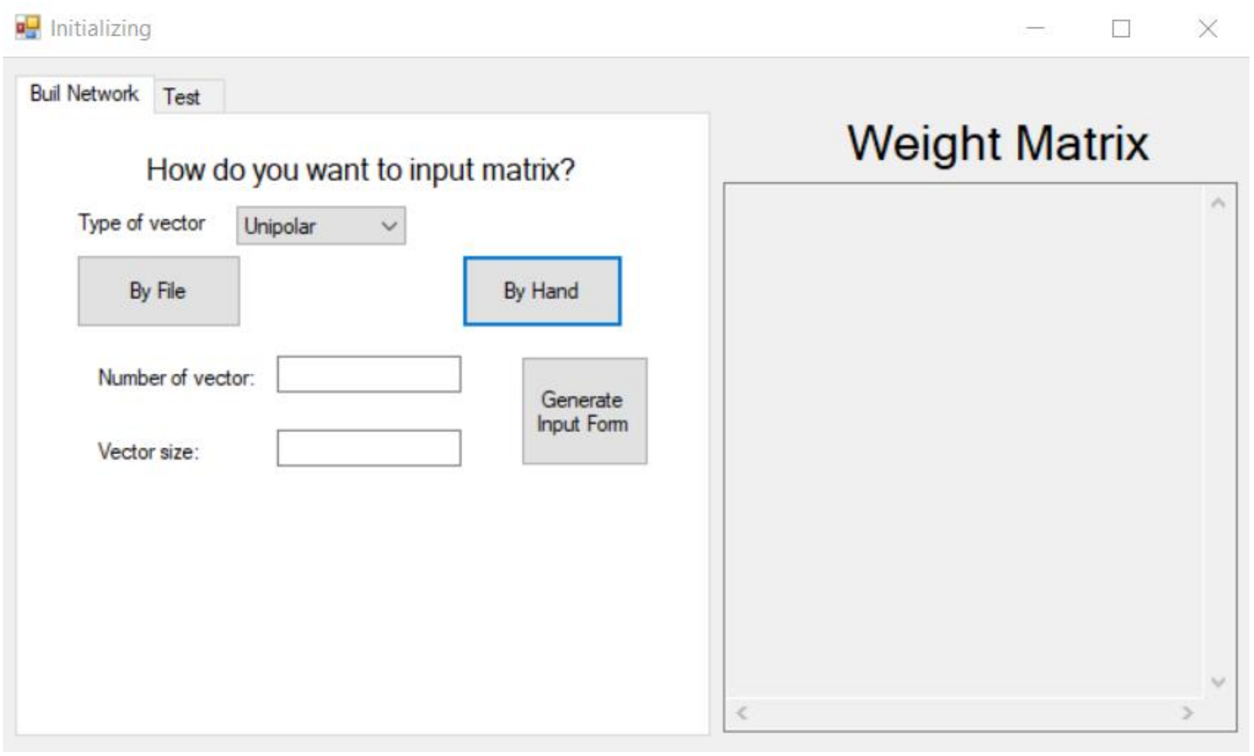
Then if user click "Build Network" weight matrix will be calculated and showed in the text box beside. Text boxes for showing file detail and weight matrix result is read only.



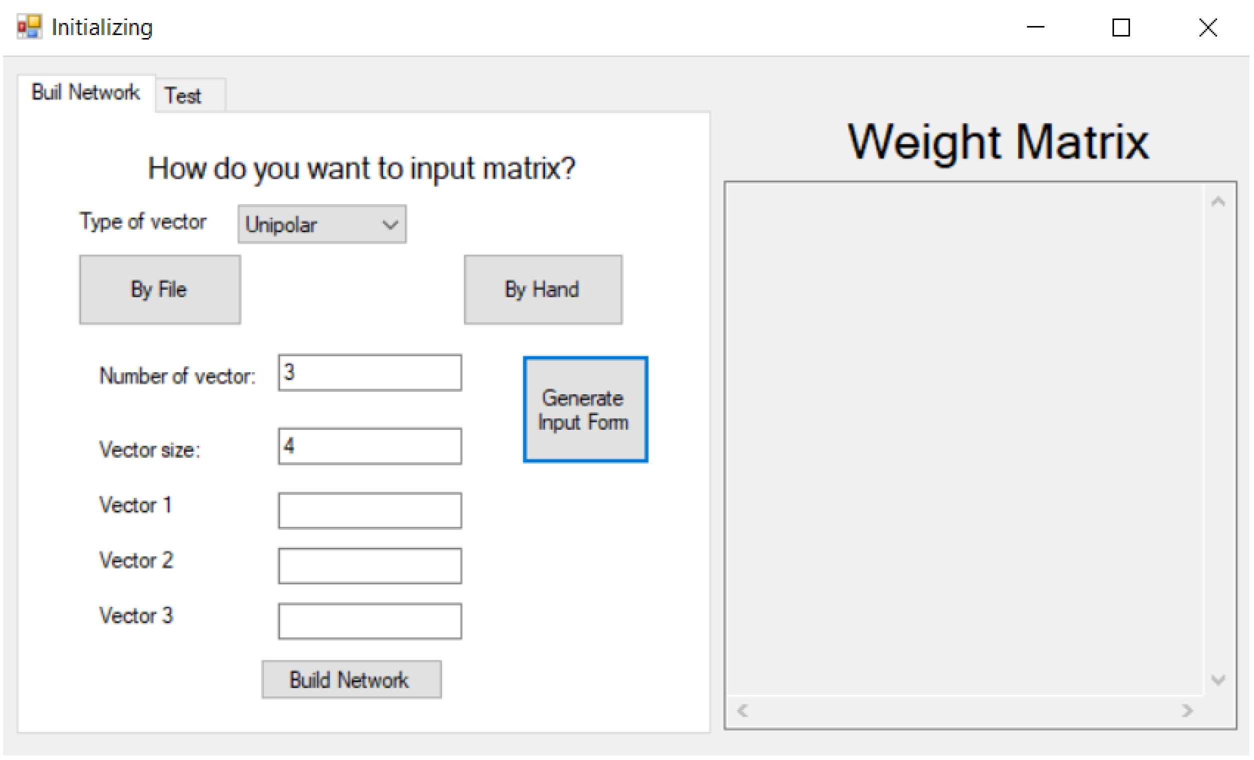
```
Input Matrix:
3
4
1 1 1 1
-1 -1 -1 -1
1 -1 -1 -1
```

```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 1 1
1 0 3 3
1 3 0 3
1 3 3 0
```

If user click "By Hand" button, the screen will display like:

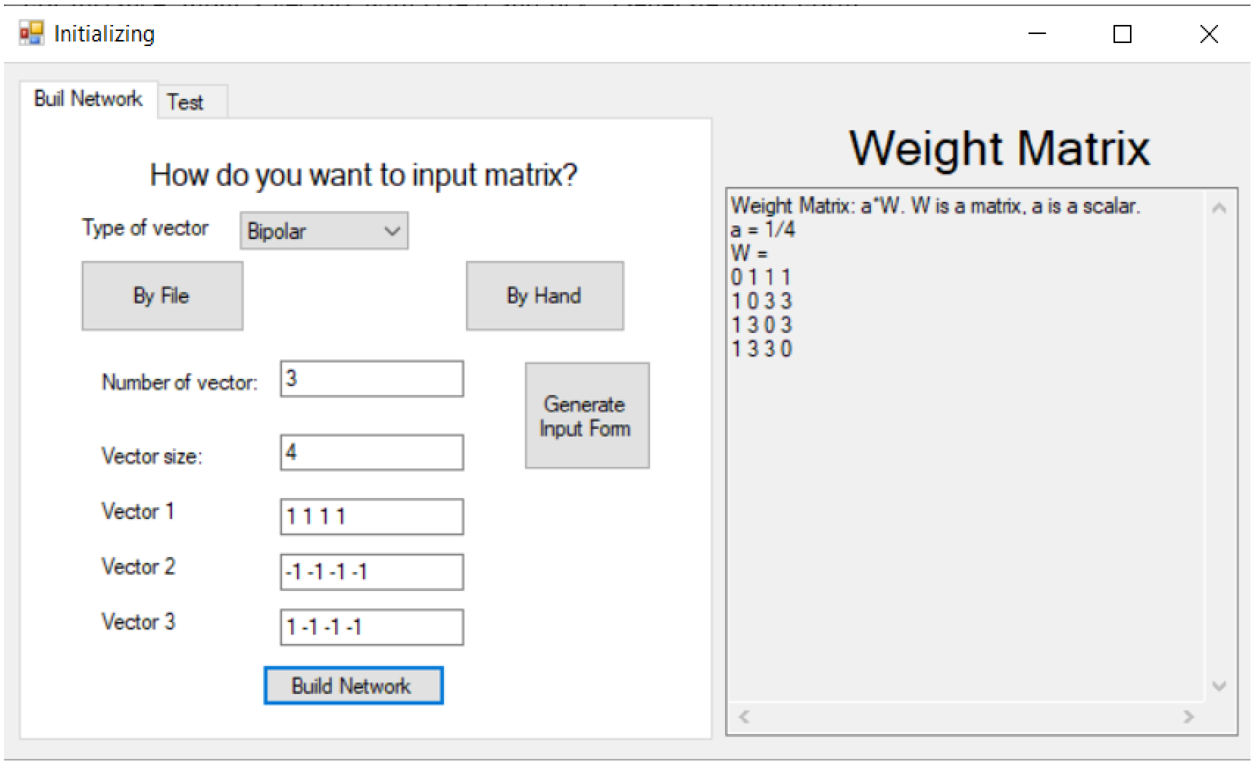


Now user has to input number of vector and vector size, then click generate input for generating form. For instance, input 3 vectors with size 4 and lick "Generate Input Form":



Type of vector: Unipolar
Number of vector: 3
Vector size: 4
Vector 1
Vector 2
Vector 3

There will be 3 text boxes for 3 vectors. User can input vector value and then click “Build Network” button for training Hopfield Network. The result will be showed in text box “Weight Matrix”.



Type of vector: Bipolar

Number of vector: 3

Vector size: 4

Vector 1: 1 1 1 1

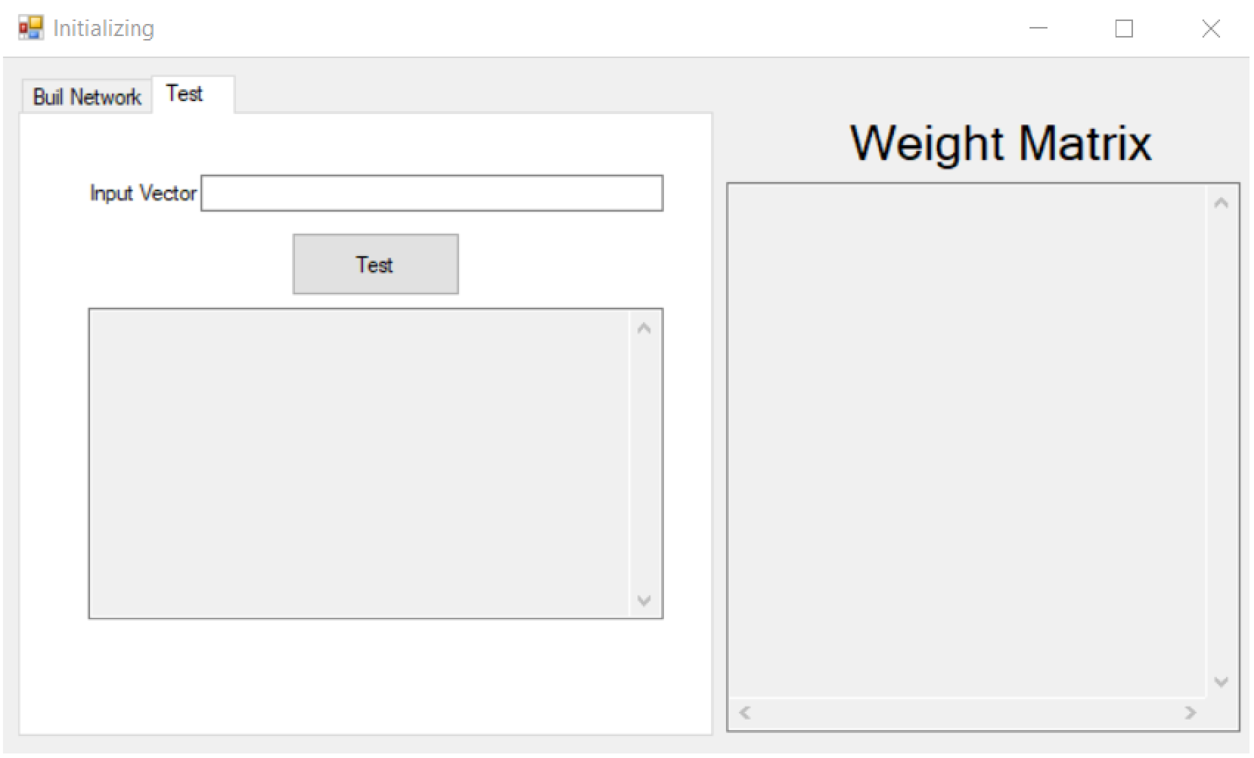
Vector 2: -1 -1 -1 -1

Vector 3: 1 -1 -1 -1

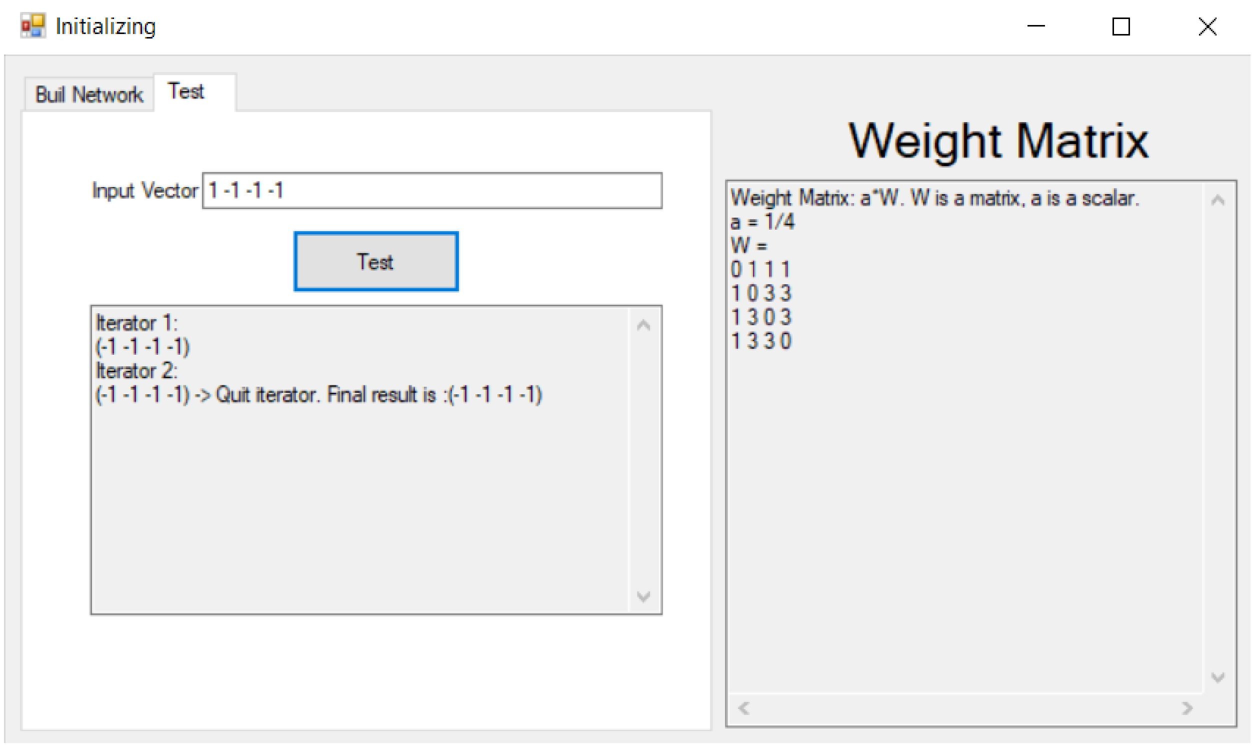
```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 1 1
1 0 3 3
1 3 0 3
1 3 3 0
```

2.2. Test Tab

The test tab GUI will be like below:



User can input vector for testing and click "Test" button. Result will be showed in text box below.



```
1 -1 -1 -1
```

```
Iterator 1:
(-1 -1 -1 -1)
Iterator 2:
(-1 -1 -1 -1) -> Quit iterator. Final result is :(-1 -1 -1 -1)
```

```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 1 1
1 0 3 3
1 3 0 3
1 3 3 0
```

The result text box will show all iterators until the final result is calculated.

3. User Guide

3.1. Input Training Vector By File

File upload has to be in .txt and follow the form: the first line is number m – number of vectors, the second line is vector size n, the next m lines are vector, each line represents vector values, which are separated by a space. Example File:

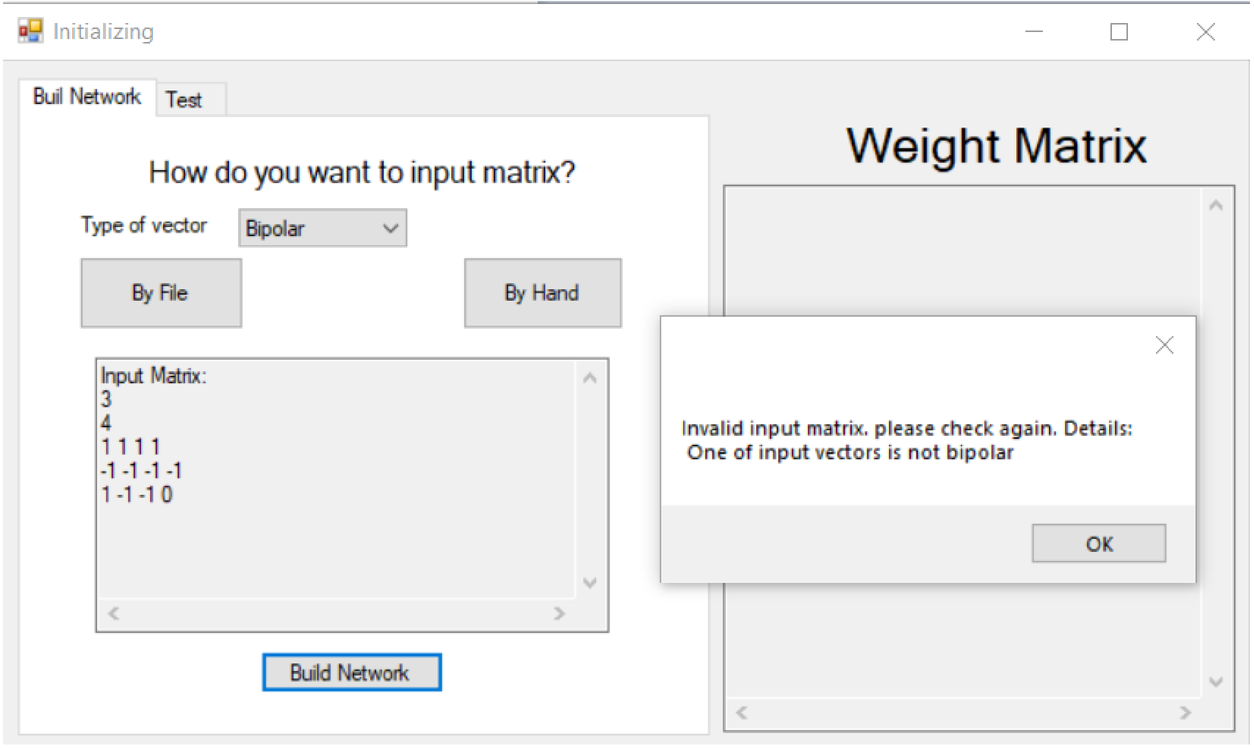
Unipolar	Bipolar

3
4
1 0 0 1
1 0 1 1
0 0 0 1
	3
	4
	1 1 1 1
	-1 -1 -1 -1
	1 -1 -1 -1

There are rules for file values:

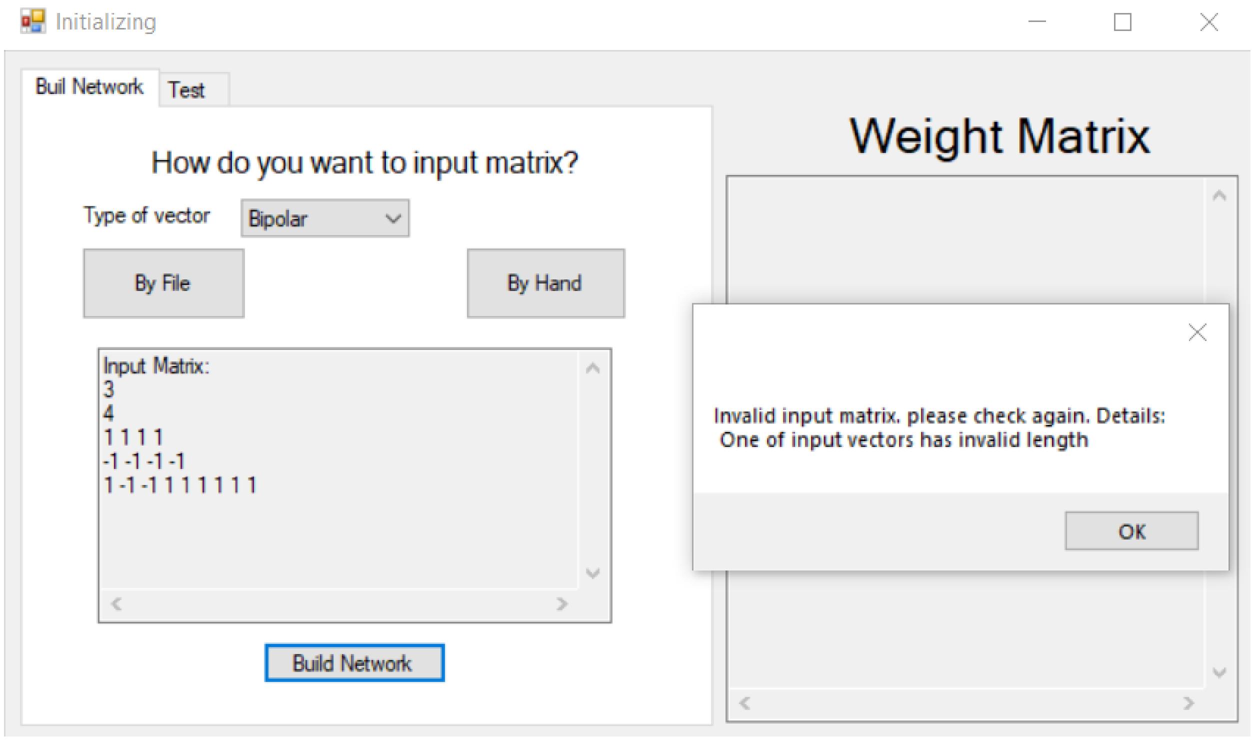
- Type of vector in file is the same with type of vector which is chose before. For example, if user choose unipolar then all the values of vector should be 0 or 1.
- In each line, the number of values equals to vector size.
- File form is the same form as described above.
- The number of lines which describe vectors equals to the number of vectors in the first line.
- All the values of vector, number of vectors and vector size are number.
- Only .txt file is supported

If all rules are passed, the network will be built. If not, the network cannot be built and a message error will be showed. Examples for error cases:



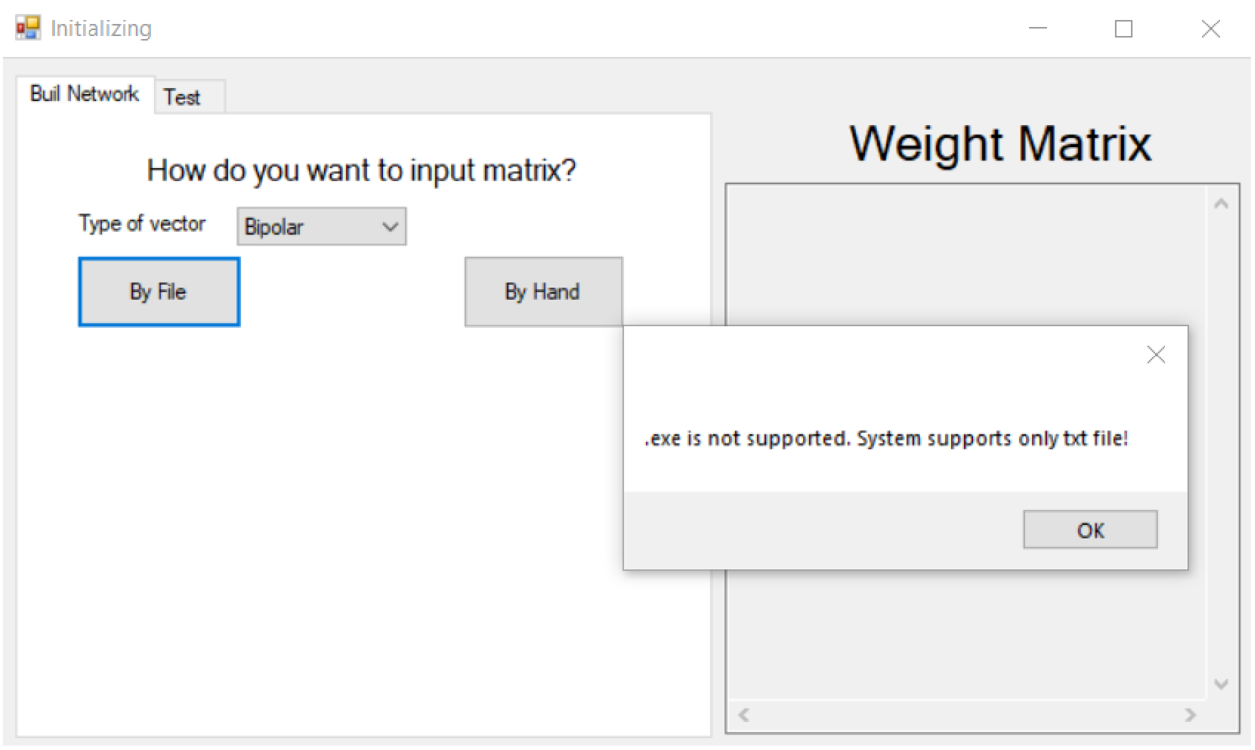
```
Input Matrix:
3
4
1 1 1 1
-1 -1 -1 -1
1 -1 -1 0
```

Invalid input matrix, please check again. Details:
One of input vectors is not bipolar



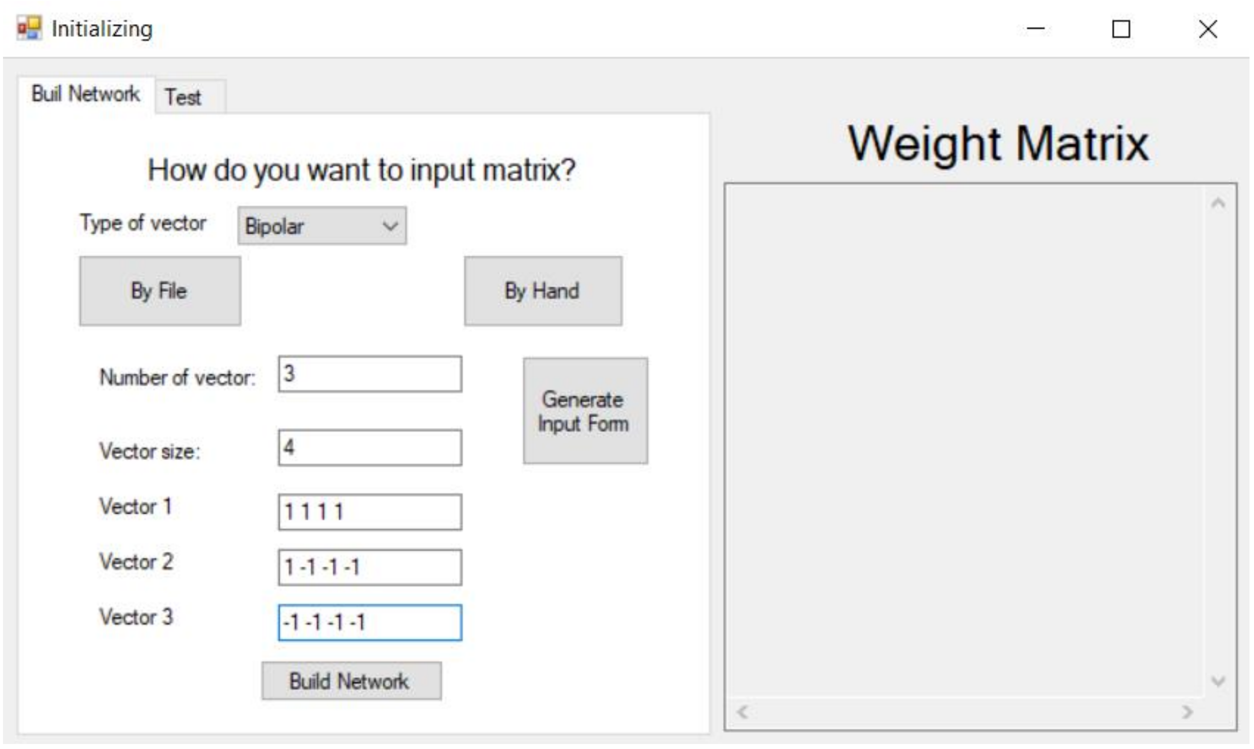
```
Input Matrix:
3
4
1 1 1 1
-1 -1 -1 -1
1 -1 -1 1 1 1 1 1 1
```

Invalid input matrix, please check again. Details:
One of input vectors has invalid length



.exe is not supported. System supports only txt file!

3.2. Input Training Vector By Hand



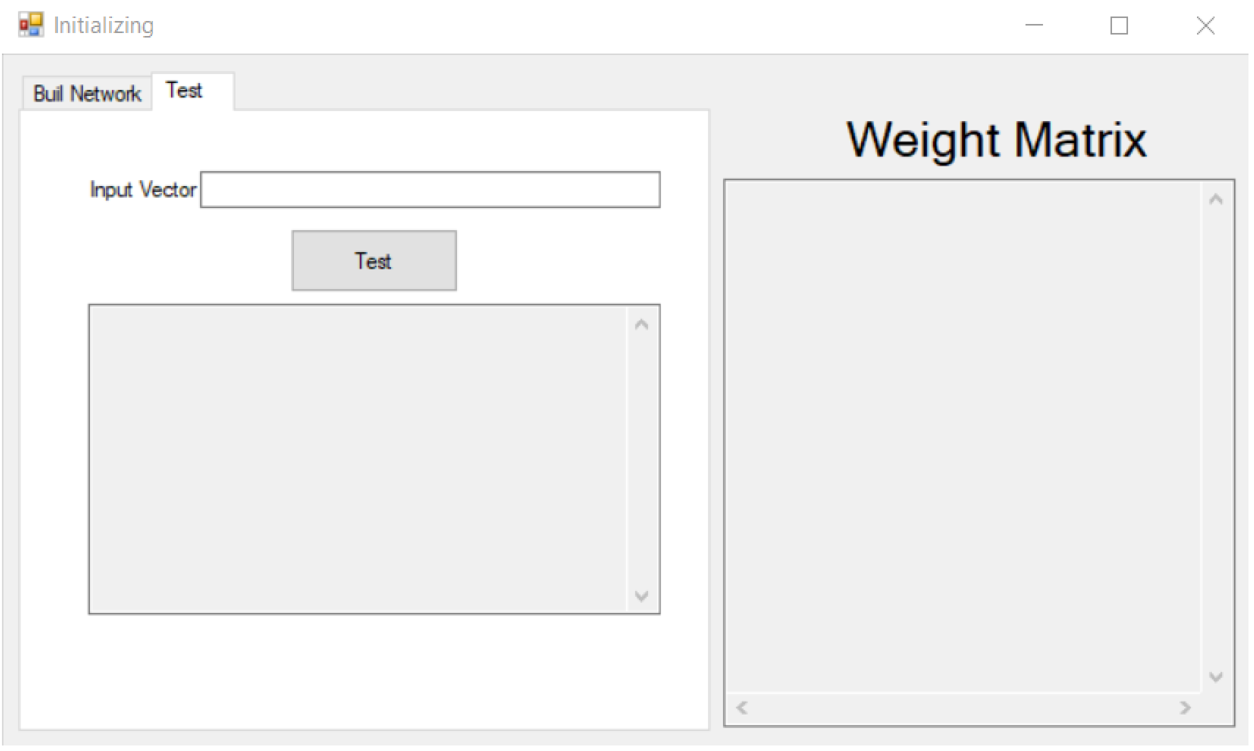
Type of vector	Bipolar
Number of vector:	3
Vector size:	4
Vector 1	1 1 1 1
Vector 2	1 -1 -1 -1
Vector 3	-1 -1 -1 -1

There are rules for inputting vectors by hand:

- Number of vector and vector size are number.
- Vector values are separated by a space.
- All values of vector are number.
- In each vector, the number of values equals to the number of values which is inputted.
- Type of vector is the same with type of vector which was chose when in training tab.

If all rules are passed, network will be built. If not, a message error will be showed, it is the same with input training vectors by uploading file.

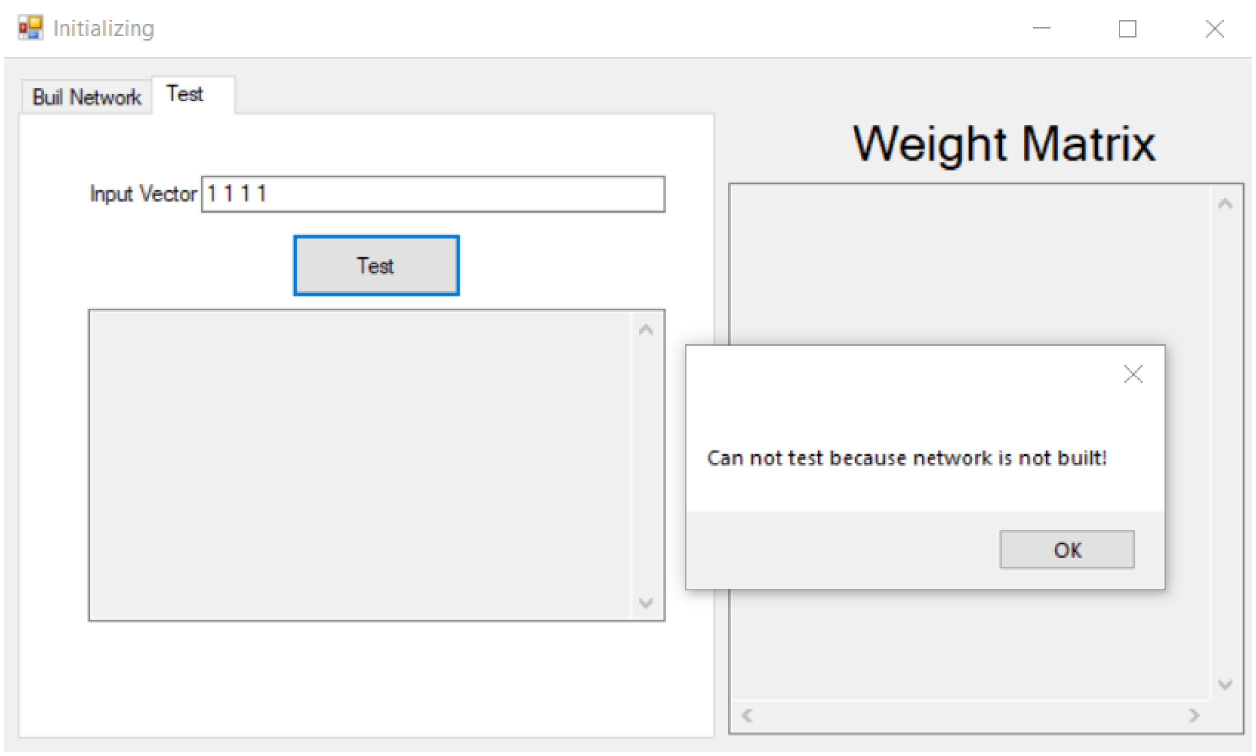
3.3. Test Vector



There are rules for testing vector:

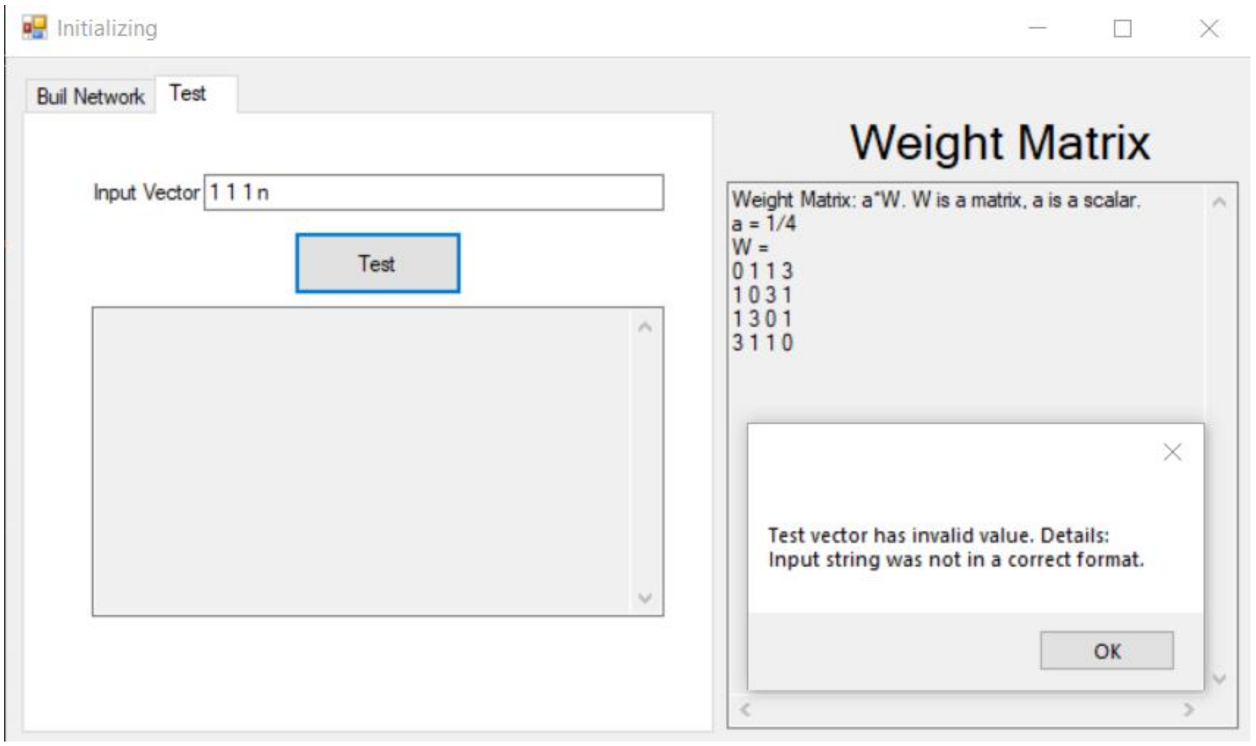
- Vector size is the same with the size which was inputted in training phase.
- Vector type is the same type with chosen type in training phase.
- All values are number and separated by a space.
- Can only test after building a network.

If all rules are passed, when click Test button, result will be calculated. If not there will be a message error. Examples for error cases:



```
1 1 1 1
```

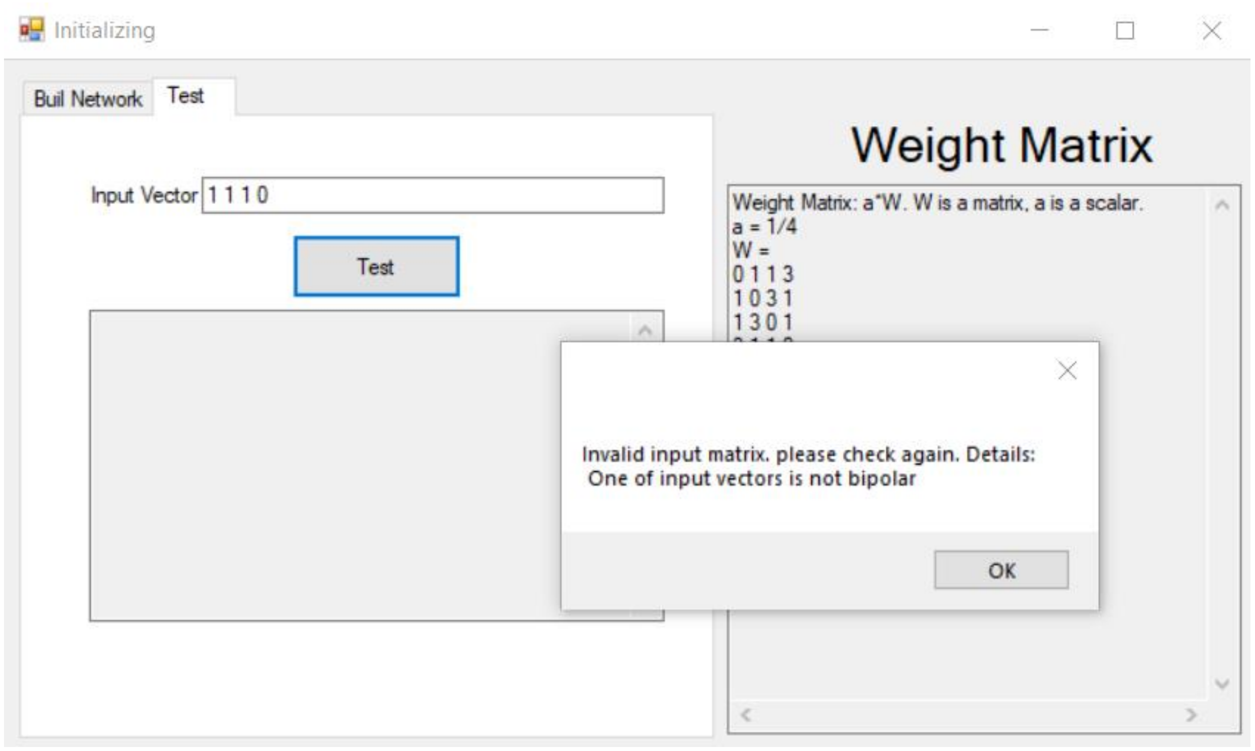
Can not test because network is not built!



```
1 1 1 n
```

```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 1 3
1 0 3 1
1 3 0 1
3 1 1 0
```

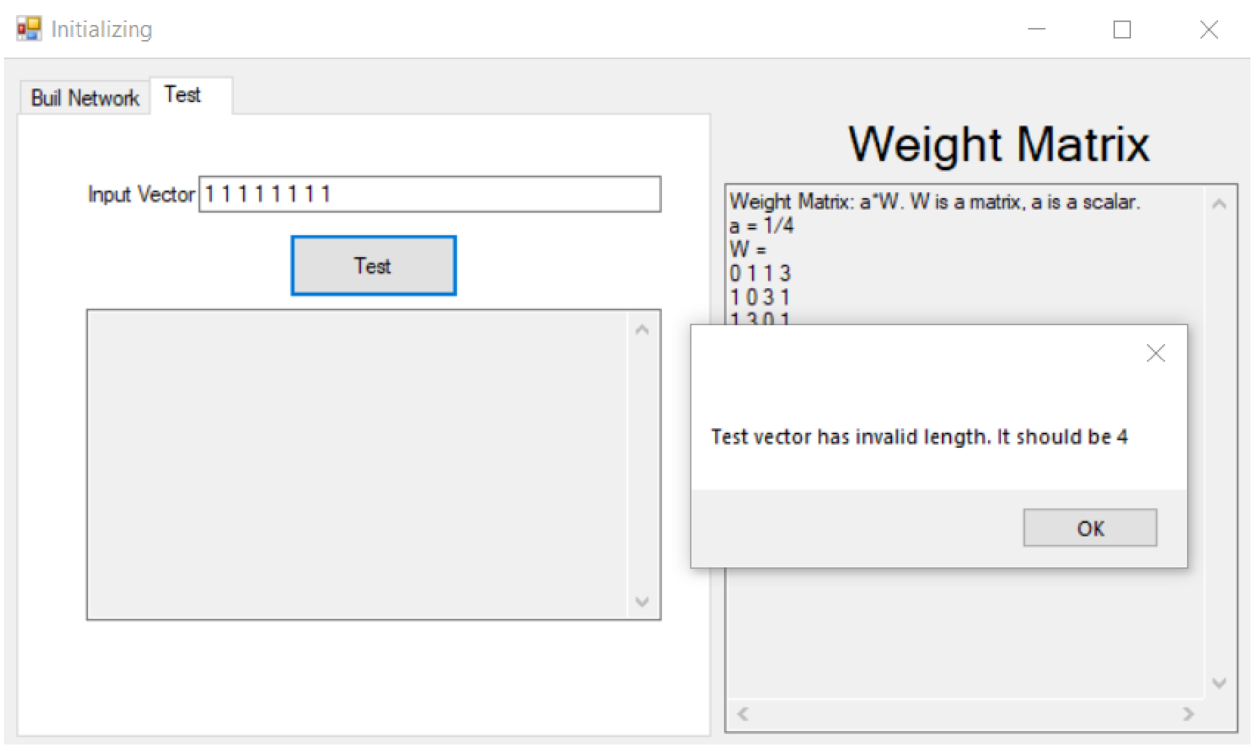
Test vector has invalid value. Details:
Input string was not in a correct format.



Input Vector 1 1 1 0

```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 1 3
1 0 3 1
1 3 0 1
```

Invalid input matrix. please check again. Details:
One of input vectors is not bipolar



Input Vector 1 1 1 1 1 1 1 1

```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 1 3
1 0 3 1
```

Test vector has invalid length. It should be 4

4. Testing

Problem: build Hopfield Associative Memory (HAM) for the following 3 vectors:

$$X_1 = [1,0,0,1]^T$$

$$X_2 = [1,1,0,1]^T$$

$$X_3 = [1,1,0,0]^T$$

4.1. Do Calculating By Hand

Step 1: this is unipolar form, transform them into bipolar form:

$$X_1 = [1,-1,-1,1]^T$$

$$X_2 = [1,-1,-1,1]^T$$

$$X_3 = [1,-1,-1,1]^T$$

Step 2: calculate weight matrix using formula (1.1), (1.2), (1.3) in section 1, we have

$$T = \frac{1}{4}\begin{bmatrix} 0 & 1 & -3 & 1 \\ 1 & 0 & -1 & -1 \\ -3 & -1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

Step 3: testing network:

Input vector V= $[1,0,0,1]^T$

Transform V to bipolar $V' = [1,-1,-1,1]^T$

$$T * V' = \frac{1}{4}\begin{bmatrix} 3 \\ 1 \\ -3 \\ 3 \end{bmatrix} -> g(T * V) = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} = V'' -> transform\ to\ unipolar \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = V''' \neq V$$

$$-> itereate\ again: T * V'' = \frac{1}{4}\begin{bmatrix} 5 \\ 1 \\ -5 \\ 1 \end{bmatrix} -> g(T * V'') = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix} -$$

$$> transform\ to\ unipolar \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = V'''$$

Quit and final result is $\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

4.2. Do Calculating By Program

Step 1: Using upload file for building network. File detail:

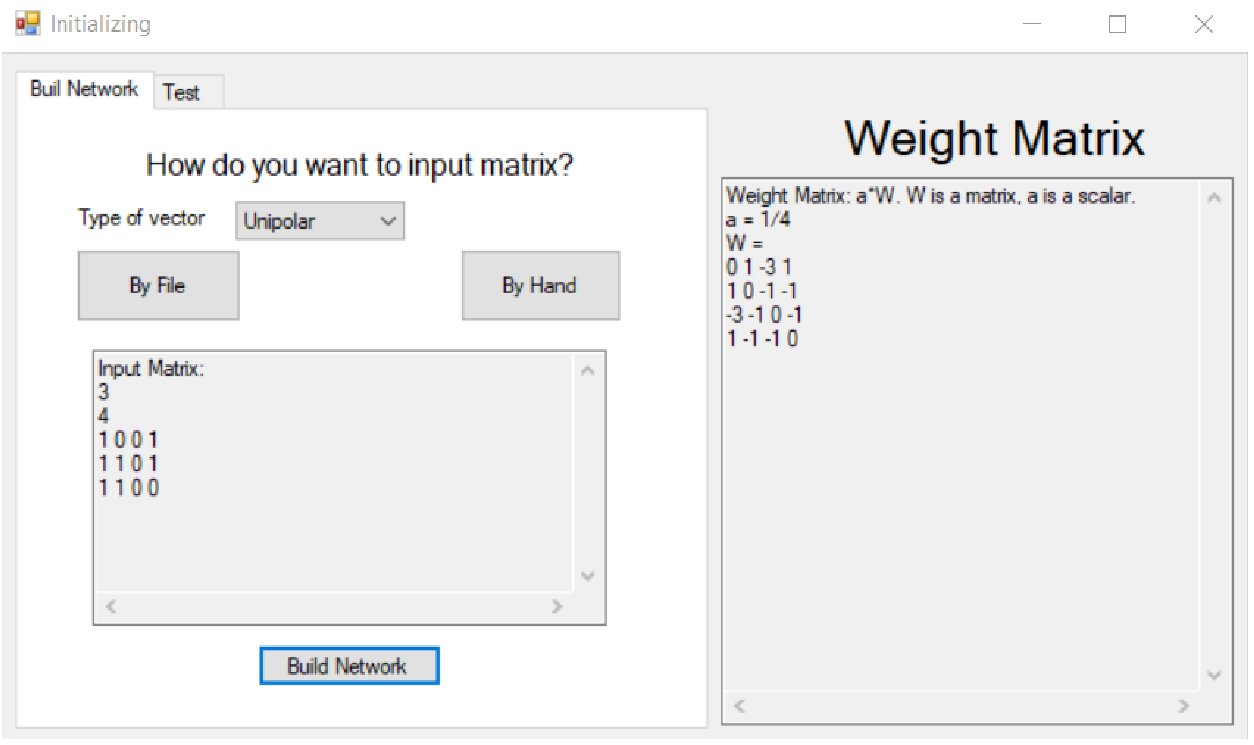
“3

4

1 0 0 1

1 1 0 1

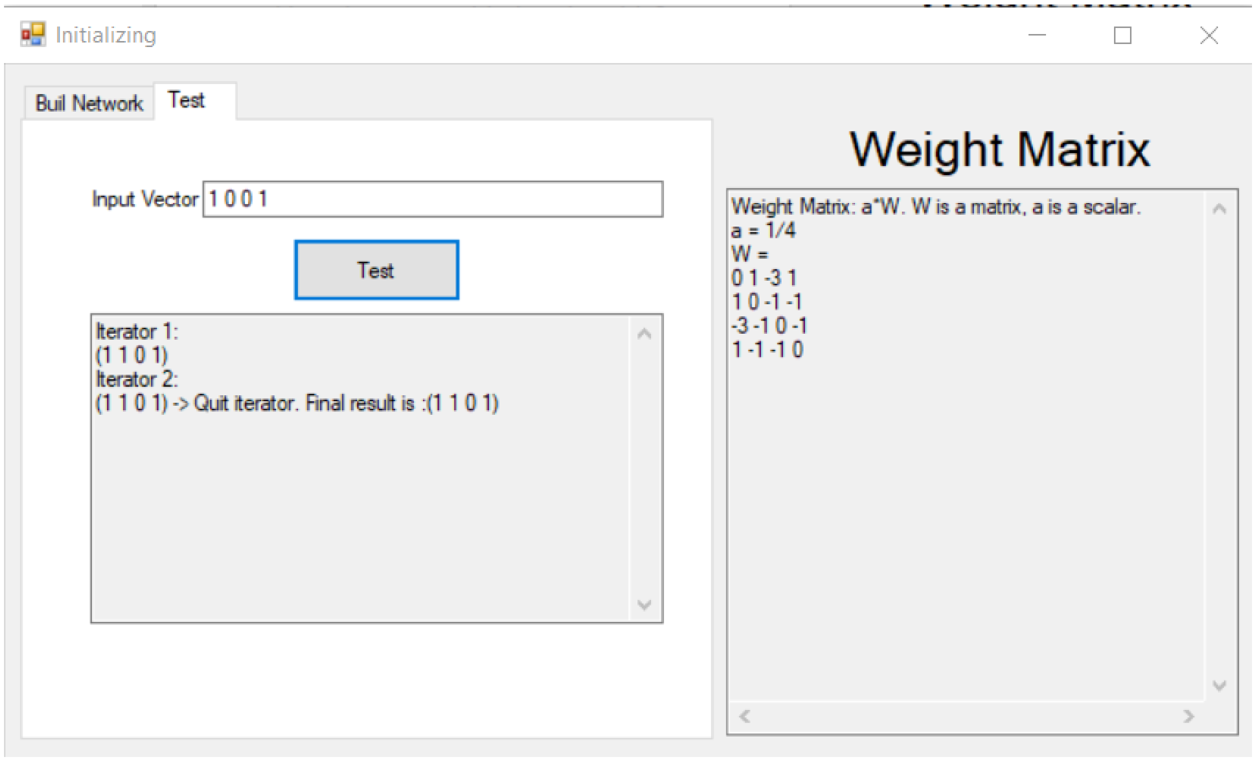
1 1 0 0”



```
Input Matrix:
3
4
1 0 0 1
1 1 0 1
1 1 0 0
```

```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 -3 1
1 0 -1 -1
-3 -1 0 -1
1 -1 -1 0
```

Step 2: testing network with input vector [1,0,0,1]



```
Iterator 1:
(1 1 0 1)
Iterator 2:
(1 1 0 1) -> Quit iterator. Final result is :(1 1 0 1)
```

```
Weight Matrix: a*W. W is a matrix, a is a scalar.
a = 1/4
W =
0 1 -3 1
1 0 -1 -1
-3 -1 0 -1
1 -1 -1 0
```

Conclude: the result of program is the same with the result when doing by hand.

“I certify that this assignment is entirely my own work, performed independently and without any help from the sources which are not allowed – Bui Tuan Anh, Student Id: 309051”