Roman Vasilyev

CISP 430

Prof Ross

4/24/2024

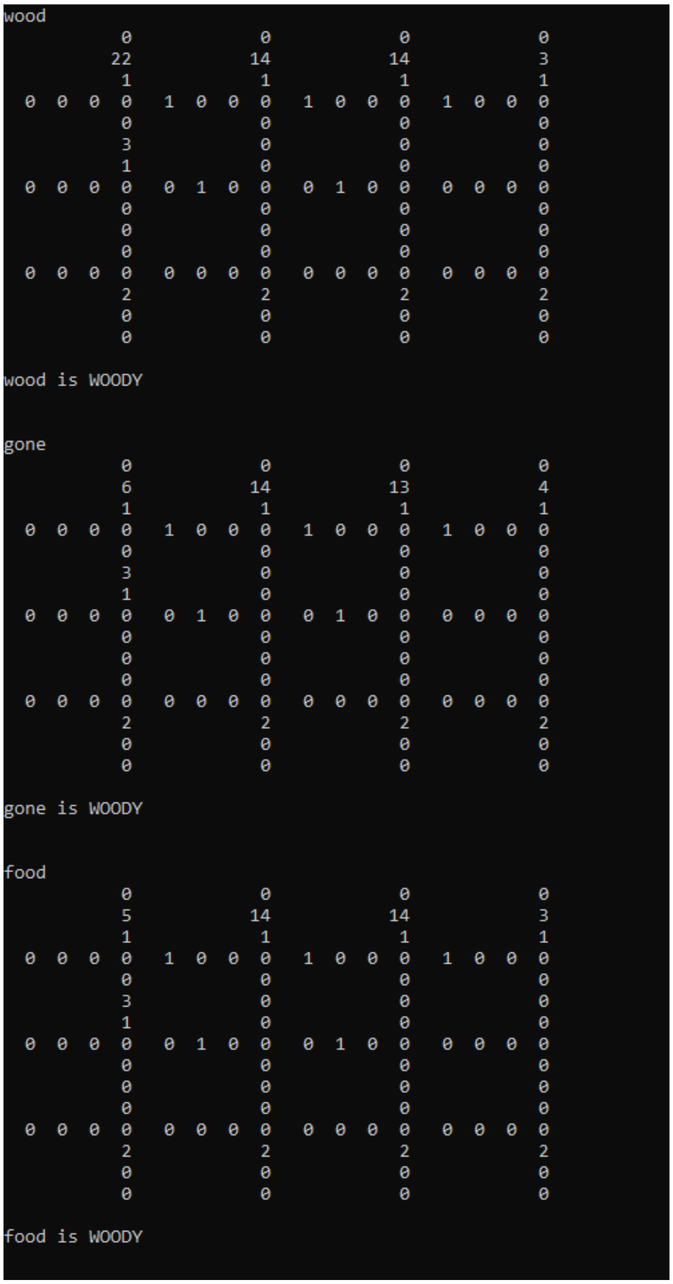
Assignment 14

```
/*
Modified Dan Ross code
Dec 2009, tweeked April 2024
*/
#include <iostream>
#include <fstream>
using namespace std;
#define N 4
#define M 4
struct node {
	int threshold; // a value above which the neuron will fire
	int effWeight; // a sum of inputs from previous layer
	int linkWeight[N]; // strength on connections to next layer (not used on last
layer)
	bool fire; // if effWeight > threshold
};
node net[M][N];
void initNet(void)
{
	// *** LAYER ZERO ***
	net[0][0].threshold = 10;
	net[0][1].threshold = 10;
	net[0][2].threshold = 10;
	net[0][3].threshold = 10;
	// *** LAYER ONE ***
	net[1][0].threshold = 10;
	net[1][1].threshold = 10;
	net[1][2].threshold = 10;
	net[1][3].threshold = 10;
	// *** LAYER TWO ***
	net[2][0].threshold = 10;
	net[2][1].threshold = 10;
	net[2][2].threshold = 10;
	net[2][3].threshold = 10;
	// *** LAYER THREE ***
	net[3][0].threshold = 10;
	net[3][1].threshold = 10;
	net[3][2].threshold = 10;
	net[3][3].threshold = 10;

	// *** LAYER ZERO ***
	net[0][0].linkWeight[0] = 1;
	net[0][0].linkWeight[1] = 1;
	net[0][0].linkWeight[2] = 1;
	net[0][0].linkWeight[3] = 1;
	net[0][1].linkWeight[0] = 1;
	net[0][1].linkWeight[1] = 1;
	net[0][1].linkWeight[2] = 1;
	net[0][1].linkWeight[3] = 1;
	net[0][2].linkWeight[0] = 1;
	net[0][2].linkWeight[1] = 1;
	net[0][2].linkWeight[2] = 1;
	net[0][2].linkWeight[3] = 1;
	net[0][3].linkWeight[0] = 1;
	net[0][3].linkWeight[1] = 1;
	net[0][3].linkWeight[2] = 1;
	net[0][3].linkWeight[3] = 1;
```

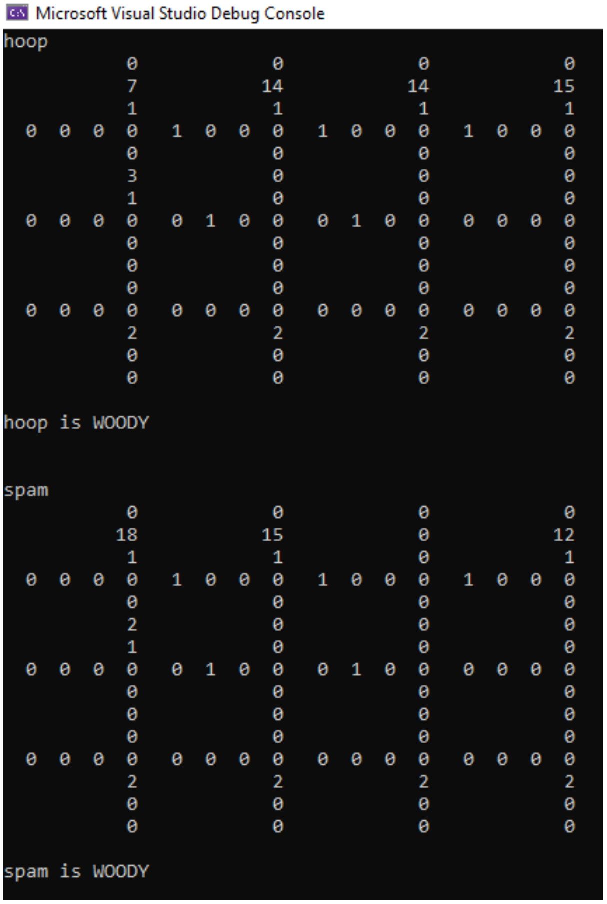
```
       // *** LAYER ONE ***
       net[1][0].linkWeight[0] = 1;
       net[1][0].linkWeight[1] = 1;
       net[1][0].linkWeight[2] = 1;
       net[1][0].linkWeight[3] = 1;
       net[1][1].linkWeight[0] = 1;
       net[1][1].linkWeight[1] = 1;
       net[1][1].linkWeight[2] = 1;
       net[1][1].linkWeight[3] = 1;
       net[1][2].linkWeight[0] = 1;
       net[1][2].linkWeight[1] = 1;
       net[1][2].linkWeight[2] = 1;
       net[1][2].linkWeight[3] = 1;
       net[1][3].linkWeight[0] = 1;
       net[1][3].linkWeight[1] = 1;
       net[1][3].linkWeight[2] = 1;
       net[1][3].linkWeight[3] = 1;
       // *** LAYER TWO ***
       net[2][0].linkWeight[0] = 1;
       net[2][0].linkWeight[1] = 1;
       net[2][0].linkWeight[2] = 1;
       net[2][0].linkWeight[3] = 1;
       net[2][1].linkWeight[0] = 1;
       net[2][1].linkWeight[1] = 1;
       net[2][1].linkWeight[2] = 1;
       net[2][1].linkWeight[3] = 1;
       net[2][2].linkWeight[0] = 1;
       net[2][2].linkWeight[1] = 1;
       net[2][2].linkWeight[2] = 1;
       net[2][2].linkWeight[3] = 1;
       net[2][3].linkWeight[0] = 1;
       net[2][3].linkWeight[1] = 1;
       net[2][3].linkWeight[2] = 1;
       net[2][3].linkWeight[3] = 1;
}
void trainNet()
{
       // Woody words: wood, gone, food, hoop, spam
       // Tinny words: tnny, meep, beek, pawn, meat
       // Manually adjust the link weights and threshold values

       // LAYER ZERO
       net[0][0].linkWeight[0] = 0; // w -> t
       net[0][0].linkWeight[1] = 0; // w -> n
       net[0][0].linkWeight[2] = 0; // w -> m
       net[0][0].linkWeight[3] = 0; // w -> p
       net[0][1].linkWeight[0] = 1; // o -> t
       net[0][1].linkWeight[1] = 0; // o -> n
       net[0][1].linkWeight[2] = 0; // o -> m
       net[0][1].linkWeight[3] = 0; // o -> p
       net[0][2].linkWeight[0] = 1; // o -> t
       net[0][2].linkWeight[1] = 0; // o -> n
       net[0][2].linkWeight[2] = 0; // o -> m
       net[0][2].linkWeight[3] = 0; // o -> p
       net[0][3].linkWeight[0] = 1; // d -> t
       net[0][3].linkWeight[1] = 0; // d -> n
       net[0][3].linkWeight[2] = 0; // d -> m
       net[0][3].linkWeight[3] = 0; // d -> p
```

```
    // LAYER ONE
    net[1][0].linkWeight[0] = 0; // t -> t
    net[1][0].linkWeight[1] = 0; // t -> n
    net[1][0].linkWeight[2] = 0; // t -> m
    net[1][0].linkWeight[3] = 0; // t -> p
    net[1][1].linkWeight[0] = 0; // n -> t
    net[1][1].linkWeight[1] = 1; // n -> n
    net[1][1].linkWeight[2] = 0; // n -> m
    net[1][1].linkWeight[3] = 0; // n -> p
    net[1][2].linkWeight[0] = 0; // n -> t
    net[1][2].linkWeight[1] = 1; // n -> n
    net[1][2].linkWeight[2] = 0; // n -> m
    net[1][2].linkWeight[3] = 0; // n -> p
    net[1][2].linkWeight[0] = 0; // n -> t
    net[1][2].linkWeight[1] = 1; // n -> n
    net[1][2].linkWeight[2] = 0; // n -> m
    net[1][2].linkWeight[3] = 0; // n -> p
    // LAYER TWO
    net[2][0].linkWeight[0] = 0; // t -> t
    net[2][0].linkWeight[1] = 0; // t -> n
    net[2][0].linkWeight[2] = 0; // t -> m
    net[2][0].linkWeight[3] = 0; // t -> p
    net[2][1].linkWeight[0] = 0; // n -> t
    net[2][1].linkWeight[1] = 0; // n -> n
    net[2][1].linkWeight[2] = 0; // n -> m
    net[2][1].linkWeight[3] = 0; // n -> p
    net[2][2].linkWeight[0] = 0; // n -> t
    net[2][2].linkWeight[1] = 0; // n -> n
    net[2][2].linkWeight[2] = 0; // n -> m
    net[2][2].linkWeight[3] = 0; // n -> p
    net[2][3].linkWeight[0] = 0; // y -> t
    net[2][3].linkWeight[1] = 0; // y -> n
    net[2][3].linkWeight[2] = 0; // y -> m
    net[2][3].linkWeight[3] = 0; // y -> p
    // LAYER THREE
    net[3][0].threshold = 2; // t
    net[3][1].threshold = 2; // n
    net[3][2].threshold = 2; // m
    net[3][3].threshold = 2; // p
}
void printnet(void)
{

    for (int row = 0; row < M; row++)
    {
        // print for each node in this layer
        for (int col = 0; col < N; col++)
        {
            cout.width(3 * N);
            cout << net[row][col].threshold << " ";
        }
        cout << endl;
        // print effective weight for each node in this layer
        for (int col = 0; col < N; col++)
        {
            cout.width(3 * N);
            cout << net[row][col].effWeight << " ";
        }
        cout << endl;
        // print fire flag for each node in this layer
        for (int col = 0; col < N; col++)
        {
            cout.width(3 * N);
            cout << net[row][col].fire << " ";
        }
        cout << endl;
        // print the weights for each node in this layer
        if (row < N - 1)
            for (int col = 0; col < N; col++)
            {
                for (int wt = 0; wt < N; wt++)
                {
                    cout.width(3);
                    cout << net[row][col].linkWeight[wt];
                }
                cout << " ";
            }
        cout << endl;
    }
}

void netIN(char* buf, int size)
{
    // feed in the ASCII value of each character

    for (int col = 0; col < size; col++)
    {
        net[0][col].effWeight = buf[col] - 'a';
        if (net[0][col].effWeight > net[0][col].threshold)
            net[0][col].fire = 1;
        else
            net[0][col].fire = 0;
    }

    int tempSum = 0;
    for (int row = 1; row <= M; row++)
    {

        for (int thisRowsCol = 0; thisRowsCol < N; thisRowsCol++)
        {
            tempSum = 0;
            for (int prevRowsCol = 0; prevRowsCol < N; prevRowsCol++)
            {
                tempSum = tempSum + net[row - 1][prevRowsCol].fire *
                    net[row - 1][prevRowsCol].linkWeight[thisRowsCol];
            }
            net[row][thisRowsCol].effWeight = tempSum;
            if (net[row][thisRowsCol].effWeight > net[row]
                [thisRowsCol].threshold)
                net[row][thisRowsCol].fire = 1;
            else
                net[row][thisRowsCol].fire = 0;
        }
    }
}
/*
```

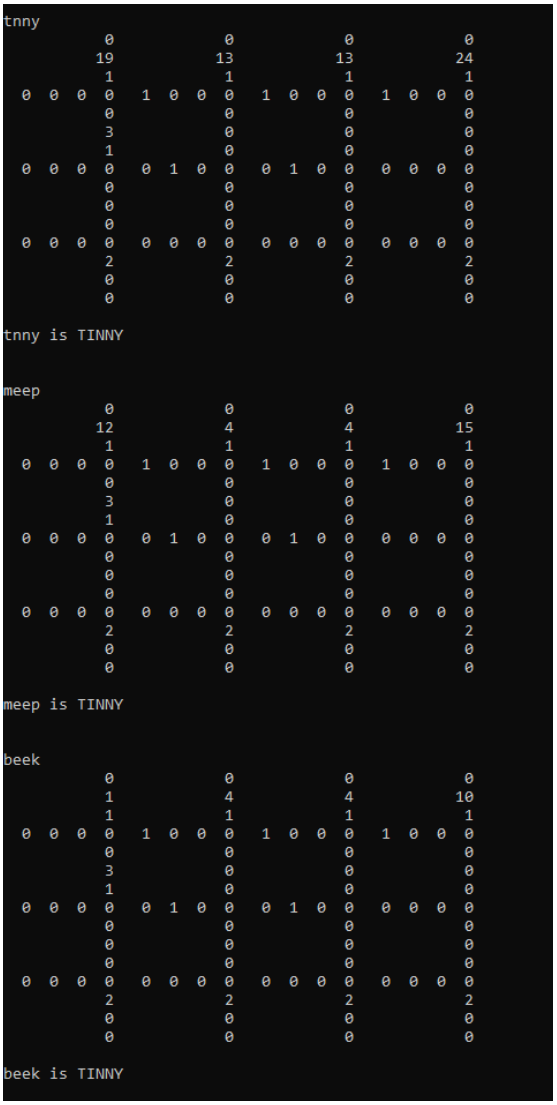
```
Greater than 7 is woody
*/
bool IsWoody(void)
{
	int value = net[3][0].fire * 8 + net[3][1].fire * 4 + net[3][2].fire * 2 + net[3][3].fire * 1;
	if (value > 7)
		return true;
	else
		return false;
}
int main(void)
{
	char word[5];
	bool woody;
	trainNet();
	// open source file
	ifstream fin("C:/Users/roman/OneDrive/Desktop/word.txt");
	if (!fin) { cout << "Input file could not be opened\n"; exit(1); }
	// loop through strings in file
	while (1) {
		fin.getline(word, 5);
		// end of file
		if (!fin) break;
		// process each word thru the net
		cout << word << endl;
		netIN(word, 4);
		printnet();
		if (IsWoody())
			cout << word << " is WOODY" << endl << endl << endl;
		else
			cout << word << " is TINNY" << endl << endl << endl;
	}
	// close file
	fin.close();
}
```



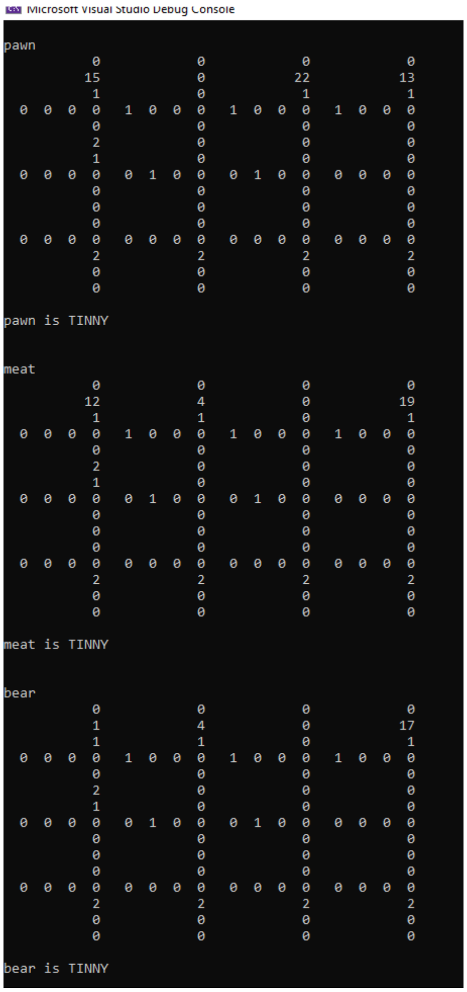
```
wood
           0            0            0            0
          22           14           14            3
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

wood is WOODY


gone
           0            0            0            0
           6           14           13            4
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

gone is WOODY


food
           0            0            0            0
           5           14           14            3
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

food is WOODY
```



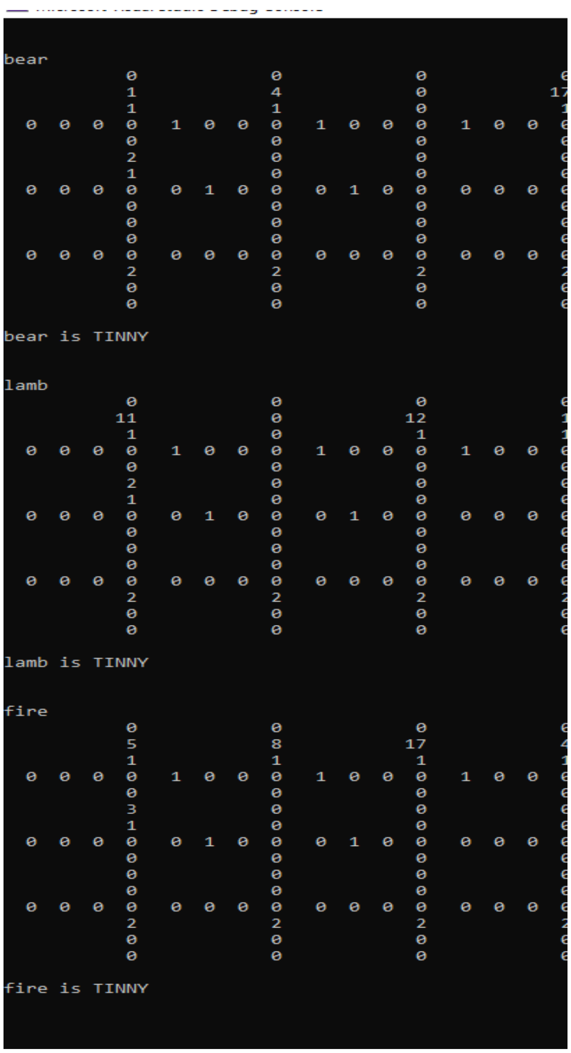
```
hoop
           0            0            0            0
           7           14           14           15
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

hoop is WOODY


spam
           0            0            0            0
          18           15            0           12
           1            1            0            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           2            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

spam is WOODY
```



```
tnny
           0            0            0            0
          19           13           13           24
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

tnny is TINNY


meep
           0            0            0            0
          12            4            4           15
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

meep is TINNY


beek
           0            0            0            0
           1            4            4           10
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

beek is TINNY
```



```

pawn
           0           0           0           0
          15           0          22          13
           1           0           1           1
  0  0  0  0  1  0  0  0  1  0  0  0  1  0  0  0
           0           0           0           0
           2           0           0           0
           1           0           0           0
  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0
           0           0           0           0
           0           0           0           0
           0           0           0           0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
           2           2           2           2
           0           0           0           0
           0           0           0           0

pawn is TINNY


meat
           0           0           0           0
          12           4           0          19
           1           1           0           1
  0  0  0  0  1  0  0  0  1  0  0  0  1  0  0  0
           0           0           0           0
           2           0           0           0
           1           0           0           0
  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0
           0           0           0           0
           0           0           0           0
           0           0           0           0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
           2           2           2           2
           0           0           0           0
           0           0           0           0

meat is TINNY


bear
           0           0           0           0
           1           4           0          17
           1           1           0           1
  0  0  0  0  1  0  0  0  1  0  0  0  1  0  0  0
           0           0           0           0
           2           0           0           0
           1           0           0           0
  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0
           0           0           0           0
           0           0           0           0
           0           0           0           0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
           2           2           2           2
           0           0           0           0
           0           0           0           0

bear is TINNY
```

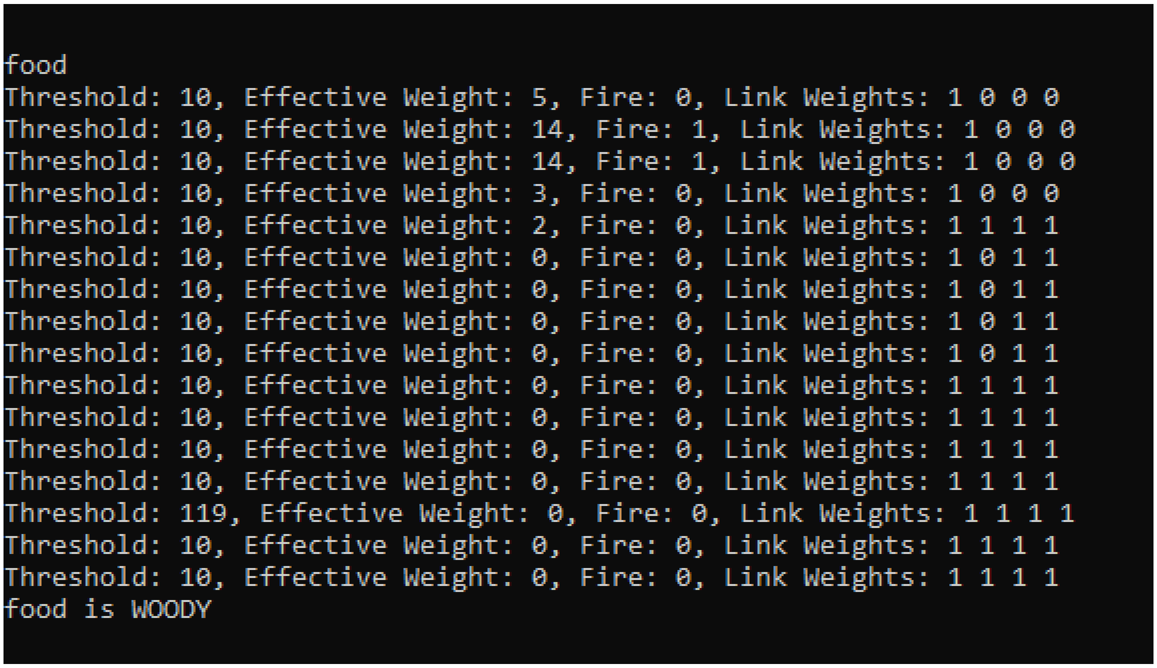


```

bear
           0            0            0            0
           1            4            0           17
           1            1            0            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           2            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

bear is TINNY


lamb
           0            0            0            0
          11            0           12            1
           1            0            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           2            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

lamb is TINNY


fire
           0            0            0            0
           5            8           17            4
           1            1            1            1
  0  0  0  0   1  0  0  0   1  0  0  0   1  0  0  0
           0            0            0            0
           3            0            0            0
           1            0            0            0
  0  0  0  0   0  1  0  0   0  1  0  0   0  0  0  0
           0            0            0            0
           0            0            0            0
           0            0            0            0
  0  0  0  0   0  0  0  0   0  0  0  0   0  0  0  0
           2            2            2            2
           0            0            0            0
           0            0            0            0

fire is TINNY
```

Part 2)

What I modified

```
void printNet() {
    // Print thresholds, effective weights, fire status, and link weights
    for (int row = 0; row < M; row++) {
        for (int col = 0; col < N; col++) {
            cout << "Threshold: " << net[row][col].threshold << ", ";
            cout << "Effective Weight: " << net[row][col].effWeight << ", ";
            cout << "Fire: " << net[row][col].fire << ", ";
            cout << "Link Weights: ";
            for (int i = 0; i < N; i++) {
                cout << net[row][col].linkWeight[i] << " ";
            }
            cout << endl;
        }
    }
```



```
food
Threshold: 10, Effective Weight: 5, Fire: 0, Link Weights: 1 0 0 0
Threshold: 10, Effective Weight: 14, Fire: 1, Link Weights: 1 0 0 0
Threshold: 10, Effective Weight: 14, Fire: 1, Link Weights: 1 0 0 0
Threshold: 10, Effective Weight: 3, Fire: 0, Link Weights: 1 0 0 0
Threshold: 10, Effective Weight: 2, Fire: 0, Link Weights: 1 1 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 0 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 0 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 0 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 0 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 1 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 1 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 1 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 1 1 1
Threshold: 119, Effective Weight: 0, Fire: 0, Link Weights: 1 1 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 1 1 1
Threshold: 10, Effective Weight: 0, Fire: 0, Link Weights: 1 1 1 1
food is WOODY
```

Part 3)

Assuming that adding two numbers consumes approximately 1 nanosecond. We encounter two nested loops that iterate over the variables "row" and "prevRowsCol." The outer loop executes "m" times, while the inner loop executes "n" times. During each iteration of the inner loop, an addition operation takes place. Consequently, we can approximate the total number of addition operations as "m * n^2."To determine the maximum value of "N" that allows the algorithm to complete within 1 second, we must find the largest "N" such that the algorithm's total execution time remains at or below 1 second. We can estimate the total execution time of the algorithm as follows: Total time = Number of addition operations * Time per addition operation = "m * n^2 * t

m * n^2 * t ≤ 1 By solving for the maximum value of "N," we can derive the following inequality:

n^2 ≤ 1 / (m * t)

n ≤ sqrt(1 / (m * t))

the largest value of "N" that allows the algorithm to complete within 1 second is approximately 5000.

Part 4)

In considering the size of a node device, a modest and straightforward machine capable of basic operations, likely compact in design is our best bet. An individual node device is estimated to have a width of 10 micrometers (10µm). The width of connecting link wires varies based on technology, design constraints, and signal needs, with a cautious estimate of 100 nanometers (100nm) for each wire. To determine the maximum number of wires, denoted as \(N \), that

can be accommodated for connecting each node, we calculate the space required for both the nodes and wires. Assuming a square layout for nodes and wires, we approximate the perimeter of a node device as four times its width. Perimeter of a node device = 4 * 10µm = 40µm. Similarly, the perimeter of each connecting link wire can be approximated as four times the width of a single wire: Perimeter of a connecting link wire = 4 * 100nm = 400nm.
To calculate the largest value of N, we need to determine the maximum number of wires that can fit within the available physical space: Perimeter of a node device * N + Perimeter of a connecting link wire * N ≤ Available space. Substituting the values: 40µm * N + 400nm * N ≤ Available space. To ensure consistent units (e.g., micrometers), we convert the values: 40 * 10^-6 * N + 400 * 10^-9 * N ≤ Available space. Simplifying the equation: 40N + 400N ≤ Available space. 440N ≤ Available space. N ≤ Available space / 440.
Now we can divide 10 mm^2 by 440 to find the maximum number of wires (N) that can be accommodated which would be .227 or 22,700