Lab 12/13

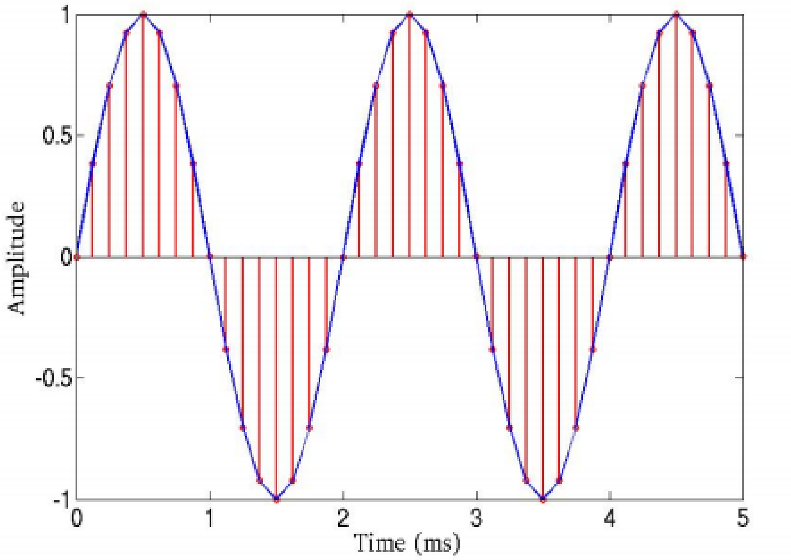
Use dynamic array or static array to do the following problems. Do not use global variables. You can use functions in math.h

■ Lab Part

1. Please display the sin signal on the screen with width=100, height=25. Let user continuously input frequency f until entering Ctrl+D.

$$y[n] = sin(\frac{2\pi f}{100}n)$$

Please construct a 2-D char array to store and plot the result. No textmode.



$$y(t) = sin(2\pi f t) \ -> sample\ at\ t = n/100 -> y[n] = sin(\frac{2\pi f}{100}n)$$

Hint:

a. Use double to calculate the proper ratio [0,1], i.e.

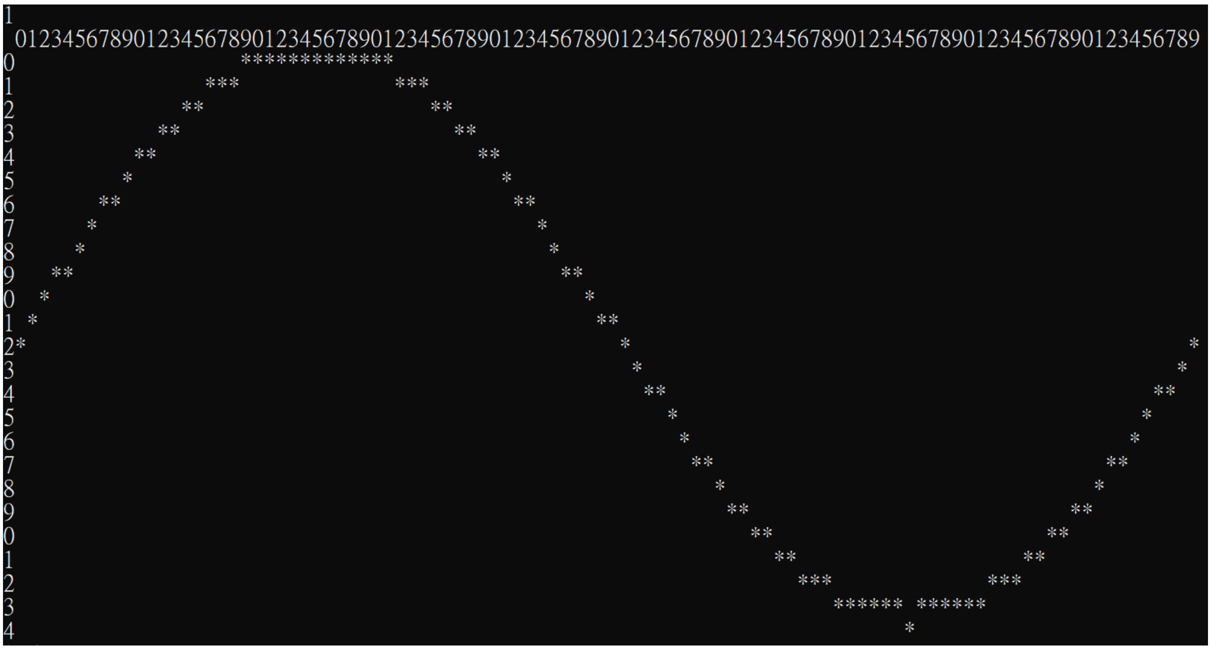
$$ratio = (y[n] - y_{min})/(y_{max} - y_{min})$$

b. transform to int index at the last step by using

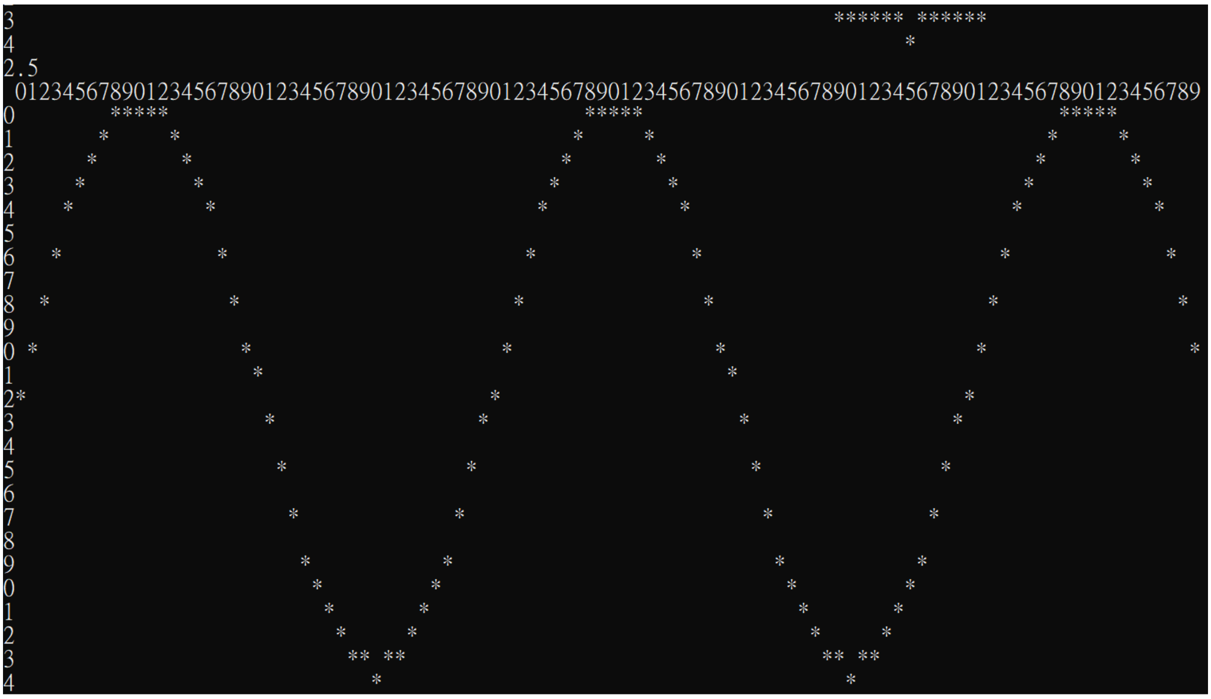
index = int(ceil(ratio*(25-1)));

thus the index will between 0 and 24, you can use it to access the element in your 2D char array

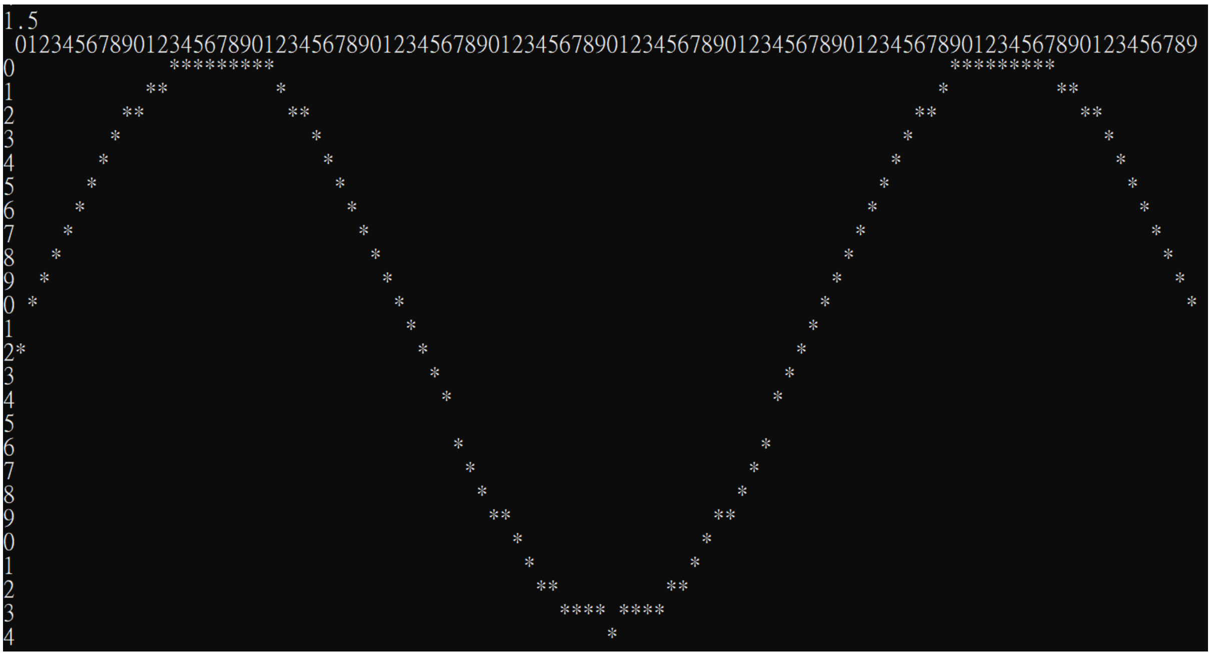
Input/Output Example:



```
1
 0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
0                   *************
1                ***             ***
2              **                   **
3            **                       **
4          **                           **
5         *                               *
6       **                                 **
7      *                                     *
8     *                                       *
9   **                                         **
0  *                                             *
1 *                                               **
2*                                                  *                                               *
3                                                    *                                             *
4                                                     **                                         **
5                                                       *                                       *
6                                                        *                                     *
7                                                         **                                 **
8                                                           *                               *
9                                                            **                           **
0                                                              **                       **
1                                                                **                   **
2                                                                  ***             ***
3                                                                     ****** ******
4                                                                           *
```



```
3                                                                     ****** ******
4                                                                           *
2.5
 0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
0        *****                                   *****                                   *****
1       *     *                                 *     *                                 *     *
2      *       *                               *       *                               *       *
3     *         *                             *         *                             *         *
4    *           *                           *           *                           *           *
5
6   *             *                         *             *                         *             *
7
8  *               *                       *               *                       *               *
9
0 *                 *                     *                 *                     *                 *
1                    *                                       *
2*                                       *                                       *
3                     *                 *                     *                 *
4
5                      *               *                       *               *
6
7                       *             *                         *             *
8
9                        *           *                           *           *
0                         *         *                             *         *
1                          *       *                               *       *
2                           *     *                                 *     *
3                            ** **                                   ** **
4                              *                                       *
```



```
1.5
 0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
0             *********                                                         *********
1           **         *                                                       *         **
2         **            **                                                   **            **
3        *                *                                                 *                *
4       *                  *                                               *                  *
5      *                    *                                             *                    *
6     *                      *                                           *                      *
7    *                        *                                         *                        *
8   *                          *                                       *                          *
9  *                            *                                     *                            *
0 *                              *                                   *                              *
1                                 *                                 *
2*                                 *                               *
3                                   *                             *
4                                    *                           *
5
6                                     *                         *
7                                      *                       *
8                                       *                     *
9                                        **                 **
0                                          *               *
1                                           *             *
2                                            **         **
3                                              **** ****
4                                                  *
```

2. Write a program to calculate Y[k], the DFT (Discrete Fourier Transform) of $y[n] = cos(\frac{2\pi n_0}{N} n)$.

$$Y[k] = \left| \sum_{n=0}^{N-1} y[n] * e^{-i\frac{2\pi}{N}*k*n} \right|, \ where\ e^{-i\theta} = cos\,(\theta) - isin(\theta)$$

$$|a + bi| = \sqrt{a^2 + b^2},\ n\,\&\,k = 0 \sim N - 1$$

Let user input frequency n_0 and sample number N until entering Ctrl+Z.

Please round Y[k] to the integer.

example: n_0 = 2, N = 8

$$Y[2] = \left| y[0] * e^{-i\frac{2\pi}{8}*2*0} + y[1] * e^{-i\frac{2\pi}{8}*2*1} + ... + y[7] * e^{-i\frac{2\pi}{8}*2*7} \right|$$

$$= \left| cos(\frac{2\pi*2}{8} * 0) * e^{-i\frac{2\pi}{8}*2*0} + cos(\frac{2\pi*2}{8} * 1) * e^{-i\frac{2\pi}{8}*2*1} + ... \right|$$

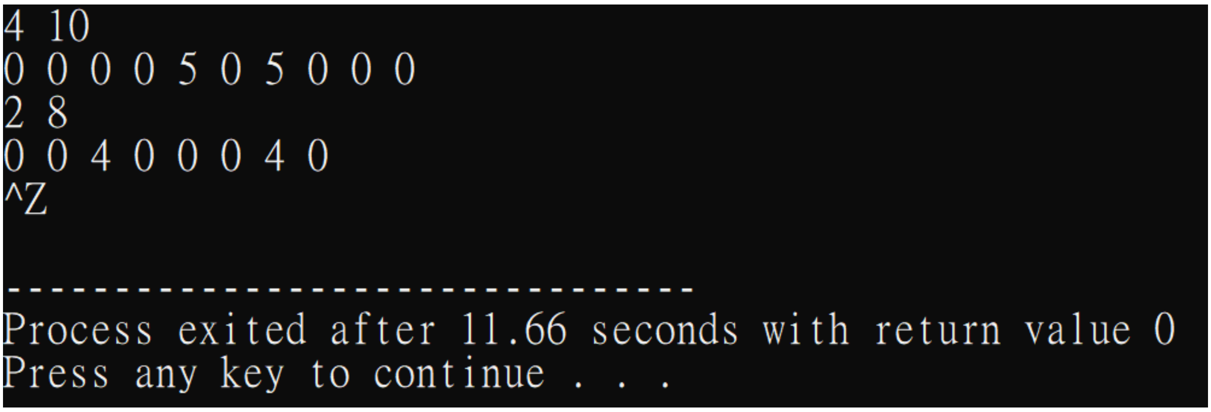
Sample input:

```
4 10
2 8
```

Sample output:

```
0 0 0 0 5 0 5 0 0 0
0 0 4 0 0 0 4 0
```

Input/Output Example:



```
4 10
0 0 0 0 5 0 5 0 0 0
2 8
0 0 4 0 0 0 4 0
^Z


--------------------------------
Process exited after 11.66 seconds with return value 0
Press any key to continue . . .
```

3. You are starting beating with the final deadline! There are so many exams/homeworks/final projects blablabla waiting for you to do :(
But you can only do one task at the same time, and you can't stop your task until you finishing it. Thus you need a good method to plan your schedule.

The waiting time of a task is defined as the time when you can start to do it. Please try to find out the minima for the average waiting time for all task.

User will first input n, which means the total number of task that waiting for you to finish, then the following is n integer number representing the total time they need to spend on. You should output the minimum average waiting time for all n tasks. The user can keep input until stopping with CTRL+Z.

Sample input:
4
1 2 3 4

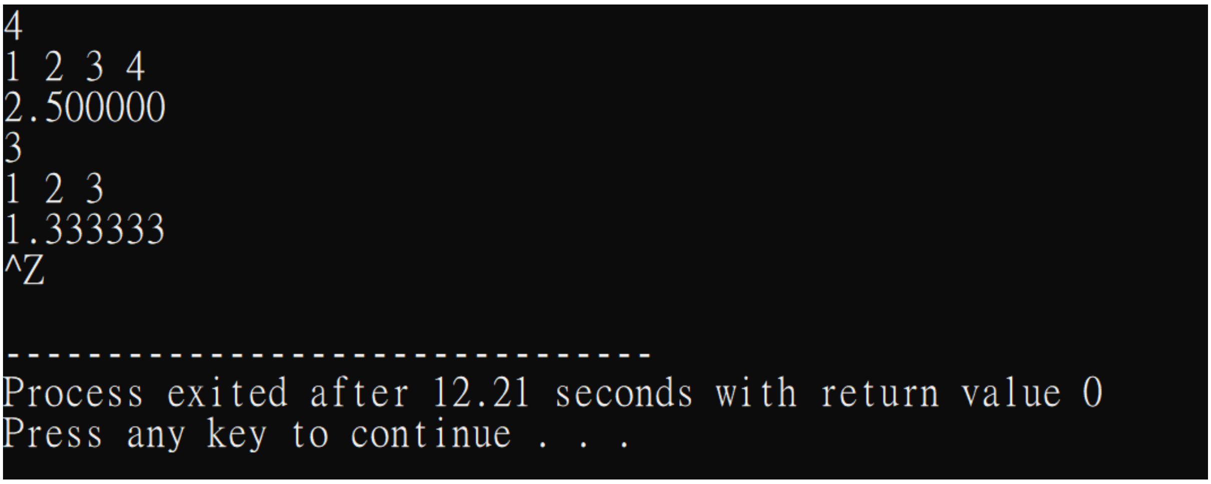
3
1 2 3

Sample output:

2.500000

1.333333

Input/Output Example:



```
4
1 2 3 4
2.500000
3
1 2 3
1.333333
^Z

--------------------------------
Process exited after 12.21 seconds with return value 0
Press any key to continue . . .
```

Homework Part

4. Design a program to simulate 100 rolls of a six-sided die:
Count the number of times each side of die appears and for each count use "*" to print out. Set the seed in srand() to 12345 initially.

Modification:

(1) six-sided die => M-sided dice

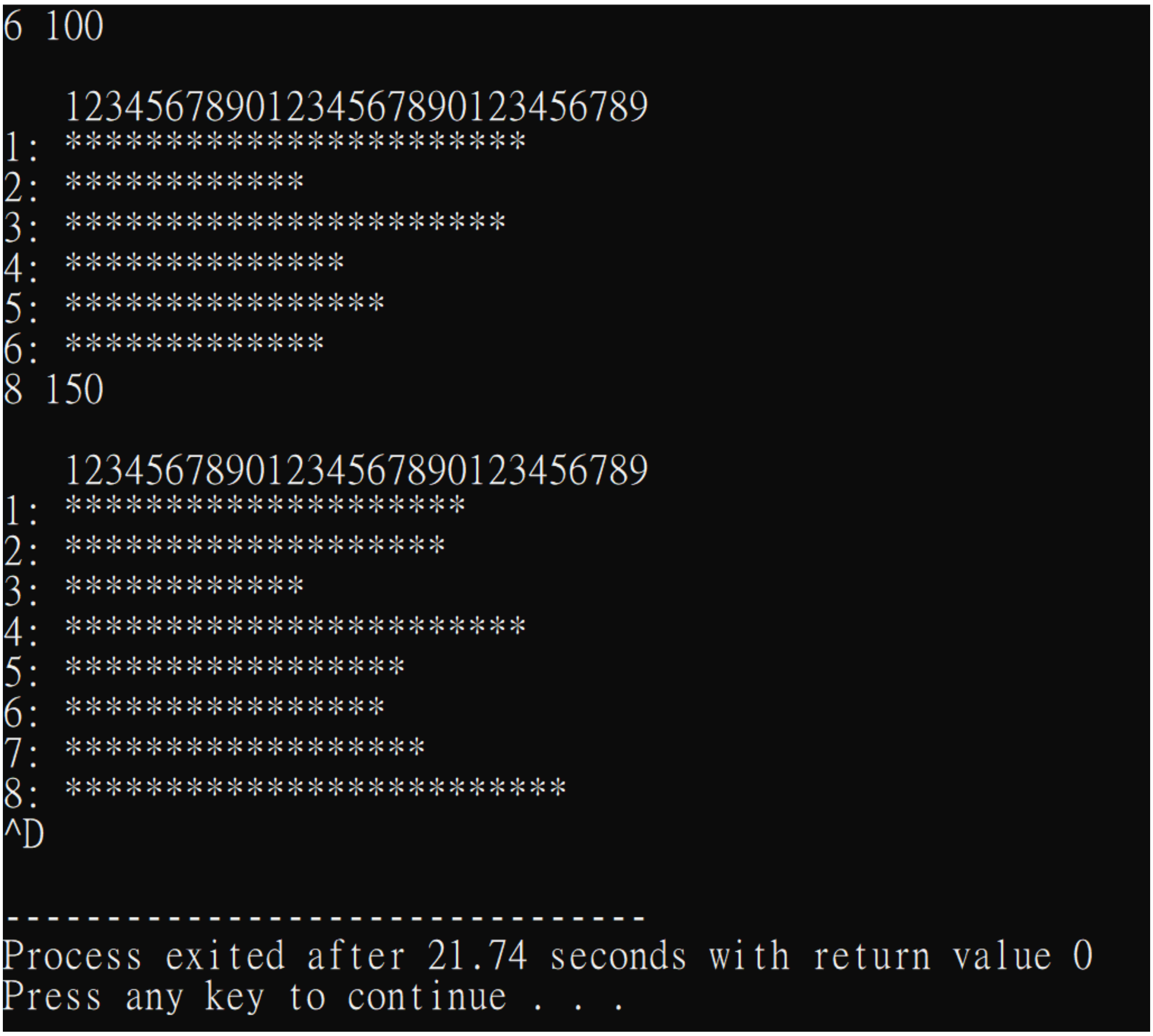
(2) simulate 100 rolls => simulate N rolls

(3) Use rand()%M to simulate the dice.

(4) Let the user continuously input M, N until inputting Ctrl+D.

(5) You cannot use switch case, please use array to implement this program.

Input/Output Example:



```
6 100

   12345678901234567890123456789
1: ***********************
2: ************
3: **********************
4: **************
5: ****************
6: *************
8 150

   12345678901234567890123456789
1: ********************
2: *******************
3: ************
4: ***********************
5: *****************
6: ****************
7: ******************
8: *************************
^D

--------------------------------
Process exited after 21.74 seconds with return value 0
Press any key to continue . . .
```

5. There are some score data in the following

72,23,38,86,62,29,99,96,67,78,87,76,65,58,86,69,93,34,45,55,52,7,74,48,
83,39,90,18,82,26,65,53,39,96,66

(a). Calculate the average score of the data.
(b). Count the number of data in each interval. The difference between each interval is 10 point. That is
0~9, 10~19, 20~29, 30~39, 40~49, 50~59, 60~69, 70~79, 80~89, 90~100
(c). Count the number of data above the average and below the average.

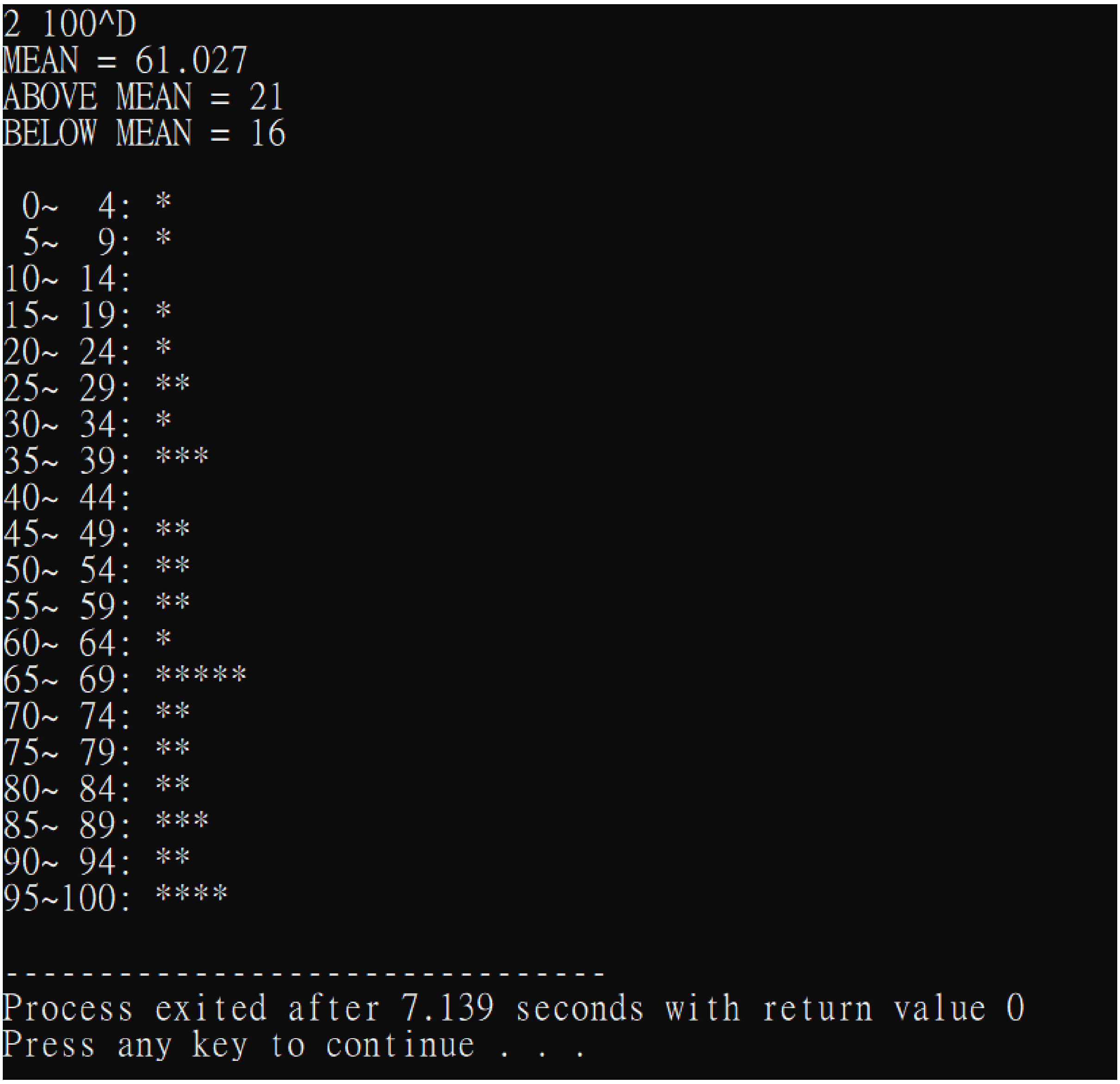
Modification:
(d) Please initialize the score data in an 1D array.

(e) Let user input additional data until Ctrl+D, total number of data will less than 100.

(f) Change the range to 0~4, 5~9, 10~14, 15~19, … , 90~94, 95~100

(g) Except the test data 100, the rest of data can be categorized by using some simple calculation +-*/%, so do not use switch case or if-else for brute force. Count with array only.

Input/Output Example:



```
2 100^D
MEAN = 61.027
ABOVE MEAN = 21
BELOW MEAN = 16

 0~  4: *
 5~  9: *
10~ 14:
15~ 19: *
20~ 24: *
25~ 29: **
30~ 34: *
35~ 39: ***
40~ 44:
45~ 49: **
50~ 54: **
55~ 59: **
60~ 64: *
65~ 69: *****
70~ 74: **
75~ 79: **
80~ 84: **
85~ 89: ***
90~ 94: **
95~100: ****


--------------------------------
Process exited after 7.139 seconds with return value 0
Press any key to continue . . .
```

6. In linear algebra, a matrix is called a Toeplitz matrix when the elements of each diagonal parallel to the main diagonal are equal between each other. For example, the following 5x5 matrix demonstrates the generic form of a 5x5 Toeplitz matrix:

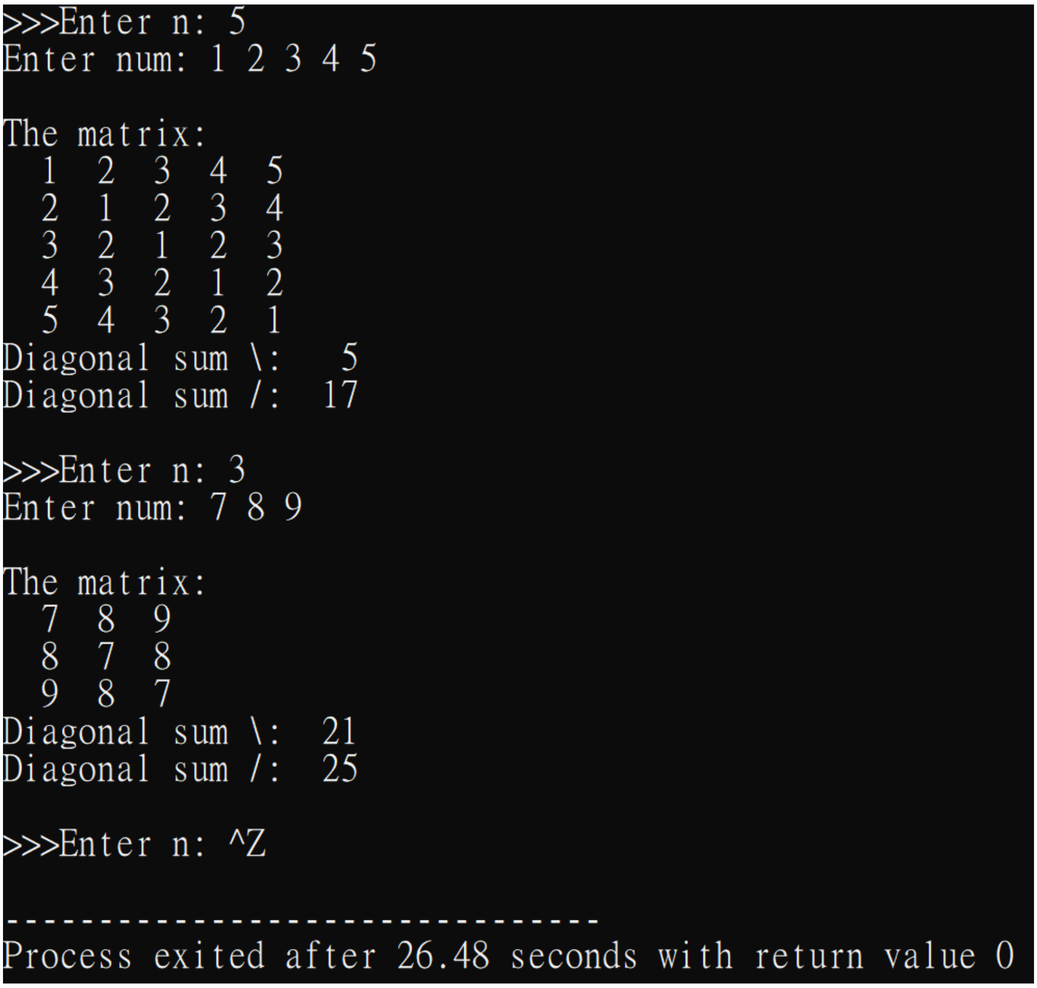
$$\mathrm{T} = \begin{bmatrix} a & b & c & d & e \\ f & a & b & c & d \\ g & f & a & b & c \\ h & g & f & a & b \\ i & h & g & f & a \end{bmatrix}$$

Write a program that reads 5 integers and stores them in the first row and first column of a 5x5 array. Next, the program should create the Toeplitz matrix and display its elements. Let the user input n, the size of n x n array, and stop the program when inputting CTRL+Z.

Modification:

(1) Also print out the diagonal sums of two direction by accessing array elements and accumulating, i.e. n*T[0][0] is not allowed .

Input/Output Example:



```
>>>Enter n: 5
Enter num: 1 2 3 4 5

The matrix:
  1  2  3  4  5
  2  1  2  3  4
  3  2  1  2  3
  4  3  2  1  2
  5  4  3  2  1
Diagonal sum \:   5
Diagonal sum /:  17

>>>Enter n: 3
Enter num: 7 8 9

The matrix:
  7  8  9
  8  7  8
  9  8  7
Diagonal sum \:  21
Diagonal sum /:  25

>>>Enter n: ^Z

--------------------------------
Process exited after 26.48 seconds with return value 0
```