Arrays

Lab 13: Processing Grades with Two Dimensional Arrays

Rung-Bin Lin

International Bachelor Program in Informatics
Yuan Ze University

Dec. 15, 2020

Arrays

- One dimensional array vs. multi-dimensional arrays
- How to declare an array?
- How to put data into arrays?
- How to pass arrays to a function?
- How to process data in arrays?

How to declare an array?

- The array size must be known upon declaration.

```
int anIntAry[10];
const int arraySize = 20;
string anStringAry[arraySize];
string aTwoDimStringAry[10][20];
const int numOfRows = 15;
const int numOfColumns = 25;
int aTwoDimIntAry[numOfRows][numOfColumns];
double aThreeDimDoubleAry[10][5][24];
```

How to initialize an array?

- Initialize it once during declaration

```
int anIntAry[5]  = {1,2,3,4,5};
string anStringAry[5] = {“1ST”, “2ND”, “3rd”}
int aTwoIntAry[4][3] ={ };
int antTwoIntAry[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
int antTwoIntAry[4][3] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

- Initialize it during execution

```
for(int i=0;i<4; i++)
      for(int j=0; j<3; j++)
            aTwoIntAry[i][j] = i+j;
```

	Column 0	Column 1	Column 2
Row 0	0	1	2
Row 1	1	2	3
Row 2	2	3	4
Row 3	3	4	5

Pass arrays to a function in a function call

- Pass by reference, so only the starting address of an array is passed.
- How to specify the starting address of an array?
  - **Use its name or the address of the first element**
    - **For one-dimensional array**

      &arrayOne[0] or arraryOne
    - **For two dimensional array**

      &arrayTwo[0][0] or arrayTwo
    - **For three dimensional array**

      &arrayThree[0][0][0] or arrayThree

Declare an array in a function's parameter list

- **Declare an array in a function prototype**

  The size of each dimension except the first dimension must be given.

```
const int dim1 = 10;
const int dim2 = 20;
const int ndim3 = 30;
void aFunc( int [ ], int x[ ], int y[ ][dim2], int [ ][dim2][dim3],
int);
```

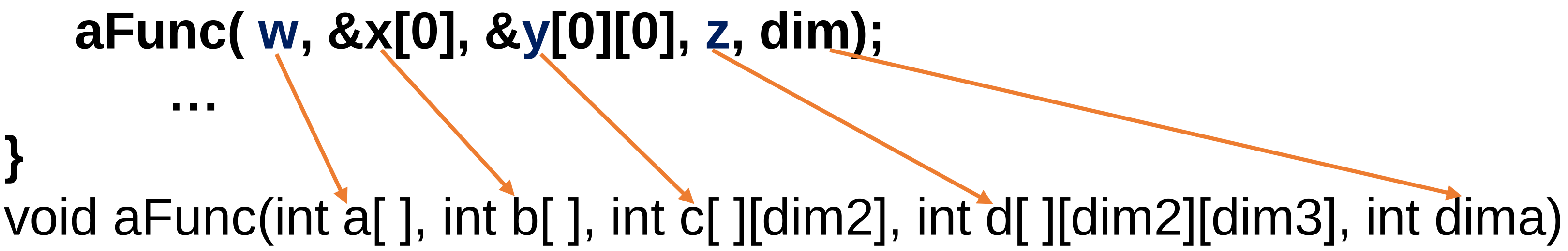
Must be constant integers

- **Declare an array in the parameter list of a function body**

```
void aFunc(int a[ ], int b[ ], int c[ ][dim2], int d[ ][dim2][dim3],
int dim1)
{
…
}
```

Make calls to a function

```
const int dim1 = 10, dim2 = 20, dim3 = 30;
void aFunc( int [ ], int x[ ], int y[ ][dim2], int [ ][dim2][dim3], int);
int main(){
    int w[28], x[dim1], y[dim1]dim2], z[dim1]dim2][dim3];
    int dim = 28
```



```
    aFunc( w, &x[0], &y[0][0], z, dim);
         …
}
void aFunc(int a[ ], int b[ ], int c[ ][dim2], int d[ ][dim2][dim3], int dima)
{
   for(int x=0; x<dima; x++)
         a[x] = x;
   for(int d1=0; d1<dima; d1++){
        b[d1] = d1+2;
        for(int d2=0; d2<dim2; d2++){
           c[d1][d2] = d1 * d2 +13;
           for(int d3=0;d3<dim3; d3++)
              d[d1][d2][d3] = d1+d2 * d3 + 23;
        }
    }
}
```

Lab 13: Processing Grades with Two Dimensional Arrays

- Problem description

Modify the code in Fig. 6.19 and do the following things:

1. **Write a function to read in the grade data from a file.**
   - *The file name should be obtained from keyboard.*
   - The file contain the grades of N students. Each student has the scores of K tests. A score is an integer and may be greater than 100. Here, K is not more than 10 and N is not more than 200.
   - The scores read in from the file should be stored in the array studentsGrades[ ][ ] declared in line 20 on page 295.
   - The function prototype for reading the file should be **void readGrades(ifstream &inFile, int studentGrades[ ][tests], int &numStudents, int &numTests);**

     The first parameter is the file object of input file. The second is the array for storing scores. The third is the *actual number of students*. The fourth is the *actual number of tests* taken by each students. You have to declare two variables for the third and fourth parameters.
   - This function should be called only once in the main() function.

Problem Description cont.

2. Modify the functions *outputGrades*, *minimum*, and *maximum* to calculate the average score of each test and the average score of each student, the minimum score of each test, and the maximum score of each test respectively. Below are hints. (40%)
   - ➢ The function *minimum*(…) should be modified to return the minimum score of a test; the function *maximum*(…) should be modified to return the maximum score of a test. This also means that the function prototype of *minimum*(…) should also include a parameter to tell which test’s minimum will be of our interest. This is same for the function prototype of *maximum*(…).
   - ➢ The output should be organized as shown in the example output. You need to modify the function *outputGrades*(…). In order to do so, you have to be able to store the average grade of each test and the average grade of each student. You can declare an array *studentGrades*[201][11] so that you can store the average grade of each test in the last row of *studentGrades* and the average grade of each student in the last column of *studentGrades*.

Problem Description cont.

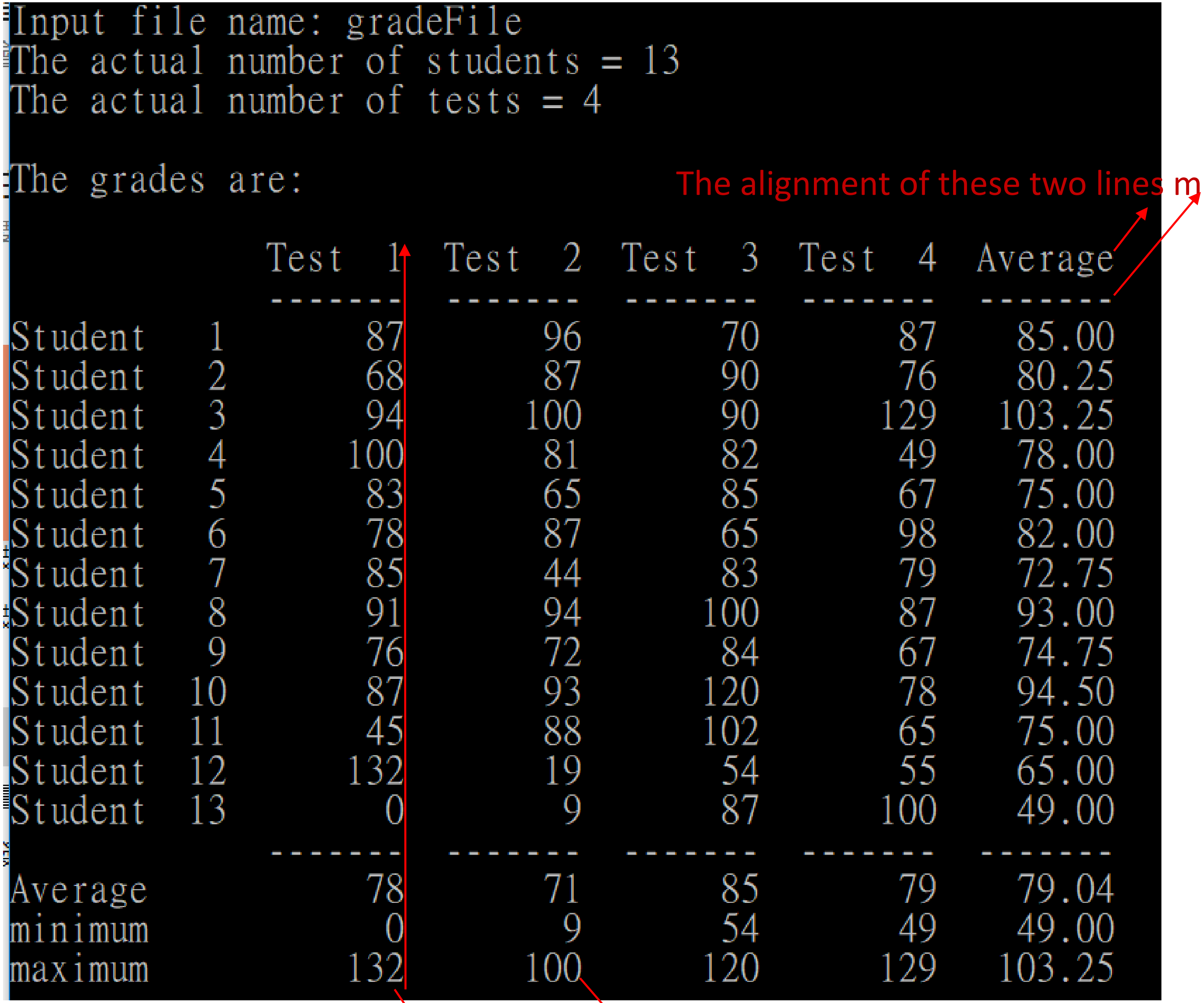
3. Modify the function *outputBarChart*(…) to print out the bar chart of the scores. If a score is larger than 100, it is classified into the group of 100. Also the bar chart should be printed as shown in the output example. Note that we have a group for zero score(30%)
4. Add a function *outputVertBarChart*(…) to print out the bar chart oriented in 90 degree clockwise and then flip over with respect to Y axis. (30%)
   - ➢ To do this, you need to check the array frequency[ ] for each category *i*. If frequency[i] is greater than zero, print a * and then reduce frequency[i] by 1. If frequency[i] is zero, we have two situations. First, if it just becomes zero, then print the original value of frequency[i]. Otherwise, print some whitespaces to prepare for printing the data of frequency[i+1]. If *i* is the index of the last element of frequency[ ], print an end-of-line.
   - ➢ Repeat the above step until all the values in frequency[ ] are zero.

Input Format

- The first line in the input file gives the actual number of students. The second line gives the actual number of tests per student. After this, each line gives the grades of all the tests of a student. Grades in each line are separated by white spaces.
- Example

```
12
4
87 96 70 87
68 87 90  76
94 100 90 129
100 81 82 49
83 65 85 67
78 87 65 98
85 44 83  79
91 94 100 87
76 72 84 67
87 93 120 78
45 88 102 65
132 19 54 55
```

Output Example



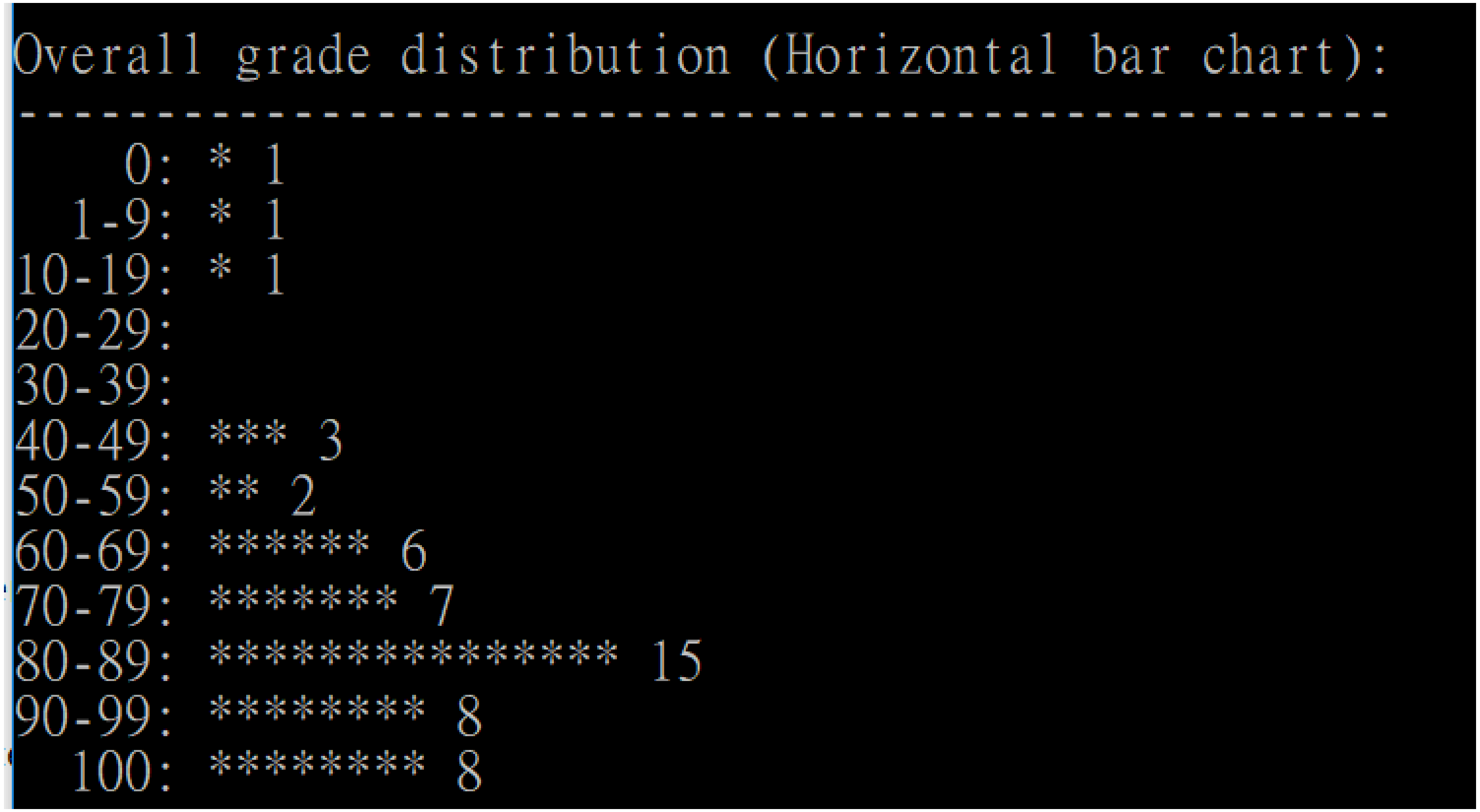
```
Input file name: gradeFile
The actual number of students = 13
The actual number of tests = 4

The grades are:

             Test   1 Test   2 Test   3 Test   4  Average
             -------- -------- -------- -------- --------
Student   1        87       96       70       87    85.00
Student   2        68       87       90       76    80.25
Student   3        94      100       90      129   103.25
Student   4       100       81       82       49    78.00
Student   5        83       65       85       67    75.00
Student   6        78       87       65       98    82.00
Student   7        85       44       83       79    72.75
Student   8        91       94      100       87    93.00
Student   9        76       72       84       67    74.75
Student  10        87       93      120       78    94.50
Student  11        45       88      102       65    75.00
Student  12       132       19       54       55    65.00
Student  13         0        9       87      100    49.00
             -------- -------- -------- -------- --------
Average            78       71       85       79    79.04
minimum             0        9       54       49    49.00
maximum           132      100      120      129   103.25
```

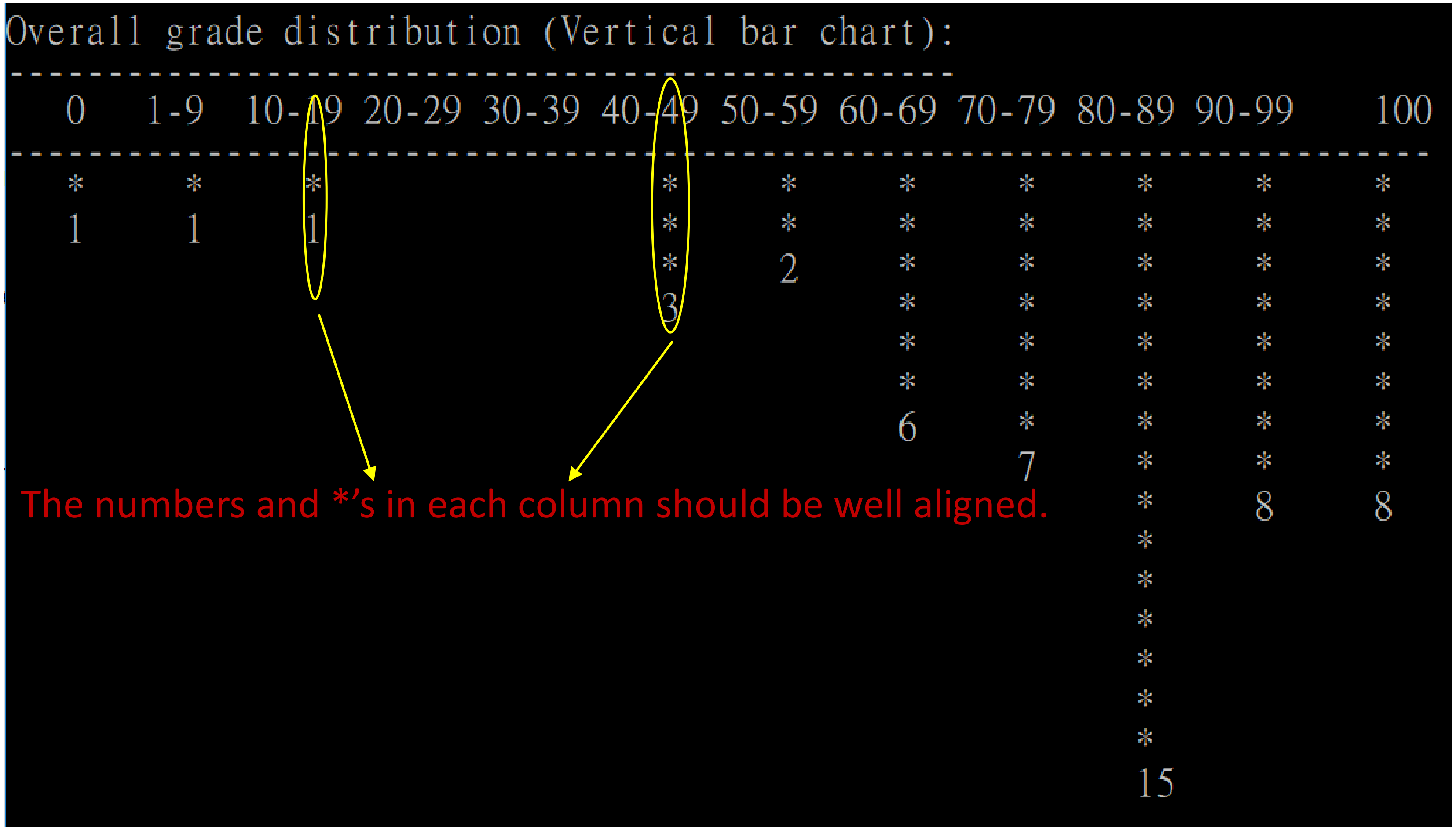
The alignment of these two lines must be correct.

The numbers in each column should be well aligned.



```
Overall grade distribution (Horizontal bar chart):
--------------------------------------------------
    0: * 1
  1-9: * 1
10-19: * 1
20-29:
30-39:
40-49: *** 3
50-59: ** 2
60-69: ****** 6
70-79: ******* 7
80-89: *************** 15
90-99: ******** 8
  100: ******** 8
```

Note that the number on the right end of each bar gives the count of each grade range.

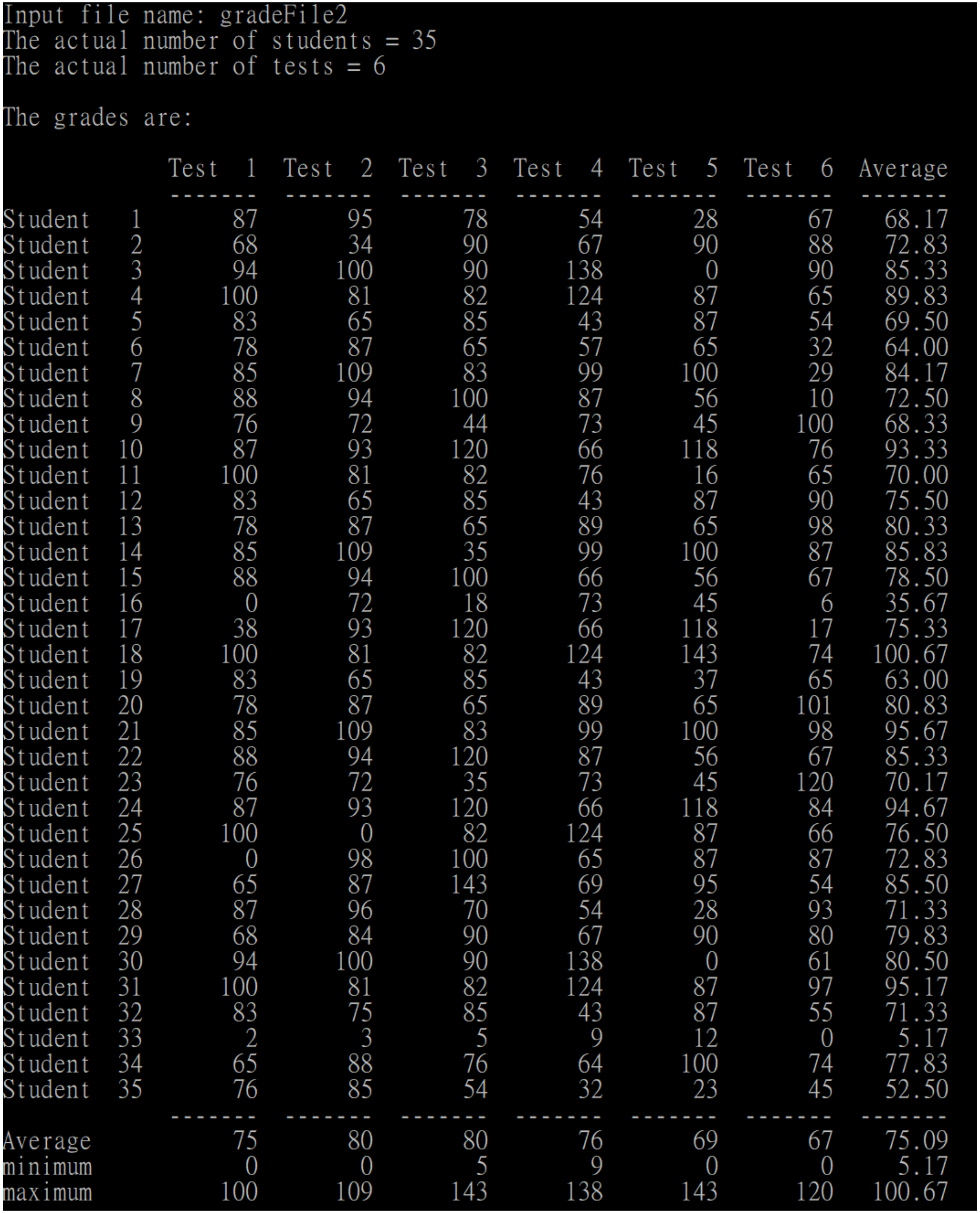


```
Overall grade distribution (Vertical bar chart):
------------------------------------------------
   0   1-9  10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99    100
------------------------------------------------------------------------
   *     *     *                 *     *     *     *     *     *     *
   1     1     1                 *     *     *     *     *     *     *
                                 *     2     *     *     *     *     *
                                 3           *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             6     *     *     *     *
                                                   7     *     *     *
                                                         *     8     8
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         15
```

The numbers and *'s in each column should be well aligned.

Note that the number at the bottom of each bar gives the count of each grade range.

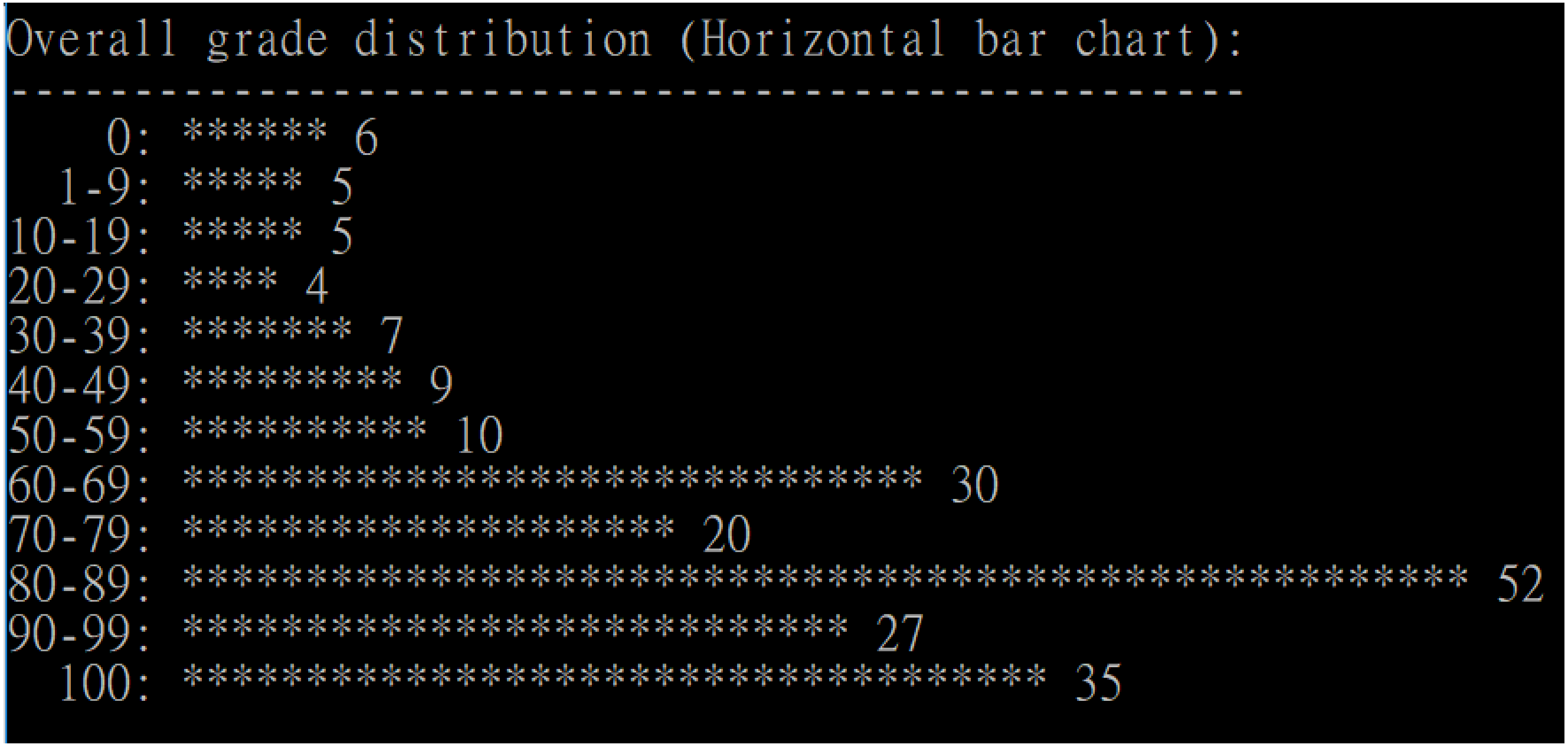
Another Example



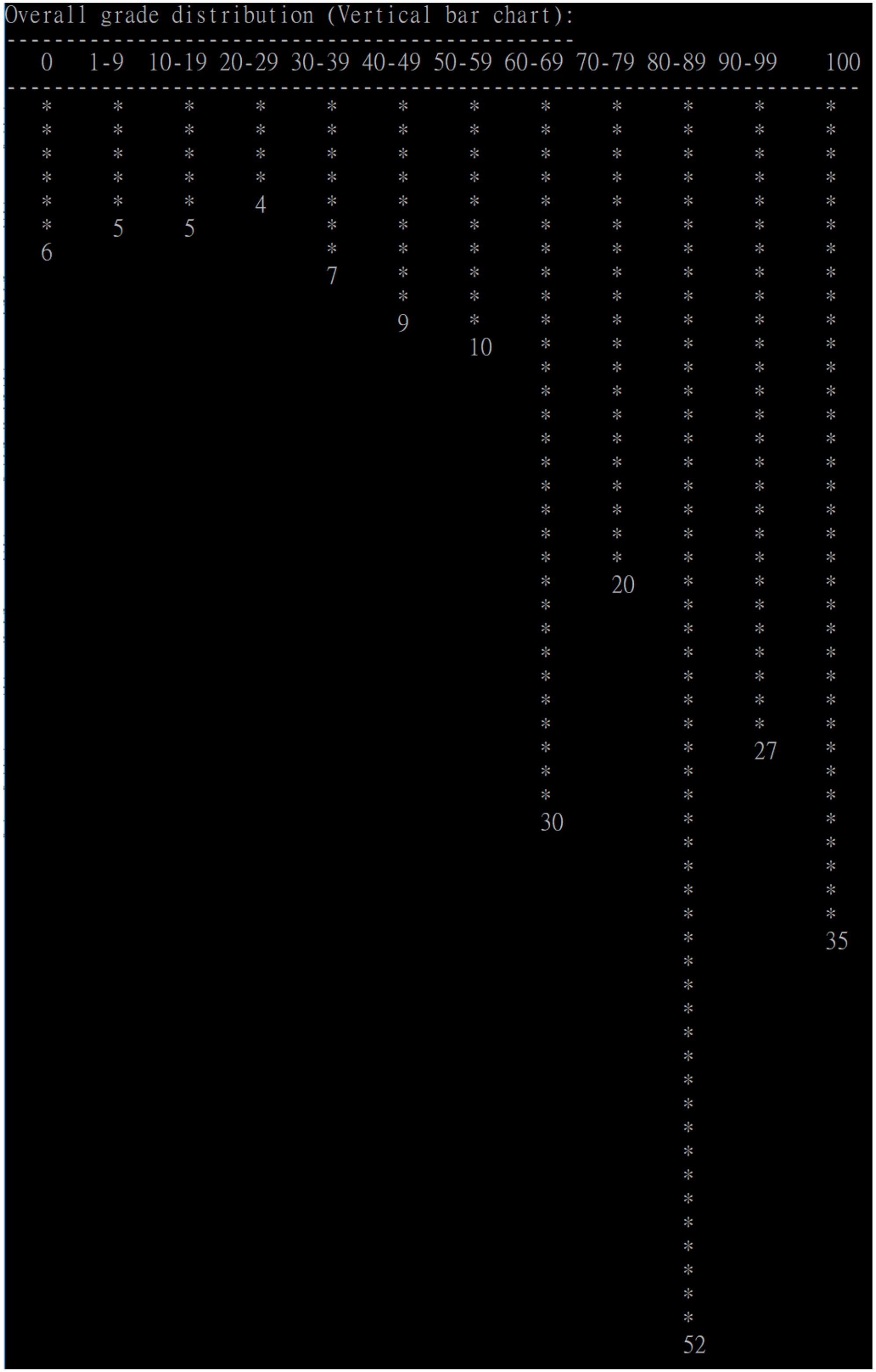
```
Input file name: gradeFile2
The actual number of students = 35
The actual number of tests = 6

The grades are:

             Test   1  Test   2  Test   3  Test   4  Test   5  Test   6   Average
              -------   -------   -------   -------   -------   -------   -------
Student   1        87        95        78        54        28        67     68.17
Student   2        68        34        90        67        90        88     72.83
Student   3        94       100        90       138         0        90     85.33
Student   4       100        81        82       124        87        65     89.83
Student   5        83        65        85        43        87        54     69.50
Student   6        78        87        65        57        65        32     64.00
Student   7        85       109        83        99       100        29     84.17
Student   8        88        94       100        87        56        10     72.50
Student   9        76        72        44        73        45       100     68.33
Student  10        87        93       120        66       118        76     93.33
Student  11       100        81        82        76        16        65     70.00
Student  12        83        65        85        43        87        90     75.50
Student  13        78        87        65        89        65        98     80.33
Student  14        85       109        35        99       100        87     85.83
Student  15        88        94       100        66        56        67     78.50
Student  16         0        72        18        73        45         6     35.67
Student  17        38        93       120        66       118        17     75.33
Student  18       100        81        82       124       143        74    100.67
Student  19        83        65        85        43        37        65     63.00
Student  20        78        87        65        89        65       101     80.83
Student  21        85       109        83        99       100        98     95.67
Student  22        88        94       120        87        56        67     85.33
Student  23        76        72        35        73        45       120     70.17
Student  24        87        93       120        66       118        84     94.67
Student  25       100         0        82       124        87        66     76.50
Student  26         0        98       100        65        87        87     72.83
Student  27        65        87       143        69        95        54     85.50
Student  28        87        96        70        54        28        93     71.33
Student  29        68        84        90        67        90        80     79.83
Student  30        94       100        90       138         0        61     80.50
Student  31       100        81        82       124        87        97     95.17
Student  32        83        75        85        43        87        55     71.33
Student  33         2         3         5         9        12         0      5.17
Student  34        65        88        76        64       100        74     77.83
Student  35        76        85        54        32        23        45     52.50
              -------   -------   -------   -------   -------   -------   -------
Average            75        80        80        76        69        67     75.09
minimum             0         0         5         9         0         0      5.17
maximum           100       109       143       138       143       120    100.67
```



```
Overall grade distribution (Horizontal bar chart):
---------------------------------------------------
    0: ****** 6
  1-9: ***** 5
10-19: ***** 5
20-29: **** 4
30-39: ******* 7
40-49: ********* 9
50-59: ********** 10
60-69: ****************************** 30
70-79: ******************** 20
80-89: **************************************************** 52
90-99: *************************** 27
  100: *********************************** 35
```



```
Overall grade distribution (Vertical bar chart):
------------------------------------------------
   0   1-9  10-19 20-29 30-39 40-49 50-59 60-69 70-79 80-89 90-99    100
------------------------------------------------------------------------
   *     *     *     *     *     *     *     *     *     *     *     *
   *     *     *     *     *     *     *     *     *     *     *     *
   *     *     *     *     *     *     *     *     *     *     *     *
   *     *     *     *     *     *     *     *     *     *     *     *
   *     *     *     4     *     *     *     *     *     *     *     *
   *     5     5           *     *     *     *     *     *     *     *
   6                       *     *     *     *     *     *     *     *
                           7     *     *     *     *     *     *     *
                                 *     *     *     *     *     *     *
                                 9     *     *     *     *     *     *
                                       10    *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     *     *     *     *
                                             *     20    *     *     *
                                             *           *     *     *
                                             *           *     *     *
                                             *           *     *     *
                                             *           *     *     *
                                             *           *     *     *
                                             *           *     *     *
                                             *           *     27    *
                                             *           *           *
                                             *           *           *
                                             30          *           *
                                                         *           *
                                                         *           *
                                                         *           *
                                                         *           *
                                                         *           35
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         *
                                                         52
```

Grading Notes for TA

- The name of the file for storing grades should be read from keyboard.
- The number of actual students and the actual number of tests should be read from the input file.
- The parameter list of readGrades() function should be **exactly the same** as that shown in void readGrades(ifstream &inFile, int studentGrades[ ][tests], int &numStudents, int &numTests);
- The count per grade range should be on the right of a horizontal bar in the horizontal bar chart.
- The count per grade range should be on the top of a vertical bar in the vertical bar chart.
- The output data should be correct.
- Other requirements are noted in the output examples.