

## Homework 1: I/O and Arrays

### **PART I:**

#### **Creating a histogram for different alphabets (A to Z) from a one dimensional character array read from the keyboard**

10 pts

#### **Specification:**

You need to write a program that reads a sentence from the keyboard, (i.e., “*The quick brown fox jumps over the lazy dog*”). Next, you will find the frequency of different alphabets (A to Z) in that sentence.

You need to declare a character array where the input from the keyboard will be read.

```
char sentence[MAXSIZE]; .....(1) Here, MAXSIZE is a constant.
```

You are given a character array of alphabets as follows:

```
char alphabets[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',  
'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' }; .....(2)
```

There are 26 alphabets. You need to have a frequency array as mentioned below. Each index of the frequency array will keep track of the number of occurrences of the alphabets as mentioned in ‘**alphabets**’ array.

```
int frequency[26]; .....(3)
```

Now, after reading a sentence from the keyboard, for each character, starting from **sentence[0]**, you need to compare which of the alphabets it falls into. For example, for the above-mentioned sentence, **sentence[0]** represents ‘**T**’. It corresponds to **alphabets[19]**. So, **frequency[19]** will be increased by one. If there is another occurrence of ‘**T**’, **frequency[19]** will again be increased. Both ‘**T**’ and ‘**t**’ will represent the same alphabet. So, you need to compare both uppercase and lowercase letter as you did in **Lab3**.

Once you finish parsing through the sentence, your frequency array will be updated. Now, for each alphabet starting from **A**, you need to print the frequency and the number of stars corresponding to frequency as captured in the following output. For example, ‘**o**’ occurred 4 times in ‘*The quick brown fox jumps over the lazy dog*’. So, 4 stars have been printed.

```

syasmin@CSCD110497WP:~/CSCD240/HW1$ ./Histogram
The quick brown fox jumps over the lazy dog
Alphabets      Frequency      Histogram
A              1              *
B              1              *
C              1              *
D              1              *
E              3              ***
F              1              *
G              1              *
H              2              **
I              1              *
J              1              *
K              1              *
L              1              *
M              1              *
N              1              *
O              4              *****
P              1              *
Q              1              *
R              2              **
S              1              *
T              2              **
U              2              **
V              1              *
W              1              *
X              1              *
Y              1              *
Z              1              *
syasmin@CSCD110497WP:~/CSCD240/HW1$

```

Please find attached the test file **Histogram\_test.c**. You need to write C code in the commented part to make the program work.

## PART II

**Performing simple arithmetic operations between two 2-D arrays.**

**10 pts**

### Specification :

You'll have two 2-D arrays of integers. A two-dimensional integer array 'a' with 4 rows and 5 column looks like the following:

```
int a[4][5] ... .. (1)
```

You'll need to fill the array elements for different array indices from the keyboard. In the example above, for each row, array '**a**' will have five elements. Starting from row 0, array index  $a[0][0]$ ,  $a[0][1]$  .....  $a[0][4]$  will fill up first, then next row starts  $a[1][0]$ ,  $a[1][1]$  ..... and so on.

So, you are going to read two 2D arrays, **a** and **b**. You need to have a function that reads array elements for different array indices from the keyboard for two arrays, i.e., **a** and **b**. Arrays **a** and **b** will have the same size, i.e., the same number of columns and rows.

Here is an example how **readArrayElements** function will look like when it reads elements for a particular array:

```
void readArrayElements(int a[][MAXCOLS], int nRows, int nCols){
    int i, j;
    for (i = 0; i < nRows; i++){
        printf("Enter data for row no: %d\n", i);

        for (j = 0; j < nCols; j++){
            scanf("%d", &a[i][j]);
        }
    }
    return;
}
```

After filling the array from keyboard input, you may want to print the array. You need to print arrays by rows. A 2D integer array **a** with two rows and two columns with the following content,  $a[0][0] = 1$ ,  $a[0][1] = 2$ ,  $a[1][0] = 3$ ,  $a[1][1] = 4$ , will print the following:

```
1 2
3 4
```

You need to write a function that prints the array. Here's the function declaration for printing an array:

```
void writeArrayElements(int a[][MAXCOLS], int nRows, int nCols); ... (2)
```

Next, you'll perform some elementwise arithmetic operations between the arrays. For example, **addArrayElements** will add elements of the corresponding indices in both arrays and will store the values in a new array **c**. For example,  $c[0][0] = a[0][0] + b[0][0]$ ,  $c[1][0] = a[1][0] + b[1][0]$ , and so on.

The resultant array **c** will have the same dimension as that of **a** and **b**. So, if **a** and **b** are declared as follows;

```
int a[2][2], b[2][2];                .... (3)
```

Array '**c**' will also be declared as follows: **int c[2][2]**.

Here's the function declaration for adding array elements:

```
void addArrayElements(int a[][MAXCOLS], int b[][MAXCOLS], int
c[][MAXCOLS], int nRows, int nCols); .....(4)
```

You'll also need to perform subtract and multiplication operations between two arrays. Subtraction of array **b** from array **a** will result in array **c**:

$c[0][0] = a[0][0] - b[0][0]$ ,  $c[1][0] = a[1][0] - b[1][0]$  and so on.

Multiplying array **a** with array **b** will result in array **c**:

$c[0][0] = a[0][0] * b[0][0]$ ,  $c[1][0] = a[1][0] * b[1][0]$  and so on.

Here are two corresponding function declarations:

```
void subtractArrayElements(int a[][MAXCOLS], int b[][MAXCOLS], int c[][MAXCOLS], int
nRows, int nCols);..... (5)

void multiplyArrayElements(int a[][MAXCOLS], int b[][MAXCOLS], int c[][MAXCOLS], int
nRows, int nCols);..... (6)
```

Now, the user will choose which array operation to perform. You need to use switch statement as follows:

```
switch (option) {
    case 'A':
    case 'a':
        printf("Adding array 'a' and 'b'\n");
        addArrayElements(a, b, c, rows, cols);
        printf("After adding array 'a' and 'b', the resultant array is: \n");
        writeArrayElements(c, rows, cols);
        break;
    case 'B':
    case 'b':
        printf("Subtracting array 'b' from array 'a'\n");
        subtractArrayElements(a, b, c, rows, cols);
        printf("After subtracting array 'b' from array 'a', the resultant array is:\n");
        writeArrayElements(c, rows, cols);
        break;
    case 'M':
    case 'm':
        printf("Multiplying array 'a' with array 'b'\n");
        multiplyArrayElements(a, b, c, rows, cols);
        printf("After multiplying array 'a' and 'b', the resultant array is:\n");
        writeArrayElements(c, rows, cols);
        break;
    default:
        printf("Invalid input\n");
}
```

So, you can see, '**A**' / '**a**' stands for addition, '**B**'/'**b**' for subtraction and '**M**'/'**m**' for multiplication.

Here's is an example how your output will look like when two arrays are multiplied:

```

syasmin@cscd-linux01:~/CSCD204$ ./Array
Enter the number of rows:2
Enter the number of columns:2

First Array:
Enter data for row no: 0
2 8
Enter data for row no: 1
3 9

Second Array:
Enter data for row no: 0
2 7
Enter data for row no: 1
3 8
Enter your option: 'A', 'B' or 'M':M
Multiplying array 'a' with array 'b'
After multiplying array 'a' with array 'b', the resulatnt array is:
  4  56
  9  72
syasmin@cscd-linux01:~/CSCD204$ █

```

Please find attached the test file **Array\_test.c**. You need to write C code in the commented part to make the program work.

### **Submission:**

- (a) Download the attached file “**Histogram\_test.c**”. Complete the code and name your C code as **Histogram.c**. Have an output capture named **Histogram\_test.pdf** – containing at least 3 different runs.
- (b) Download the attached file “**Array\_test.c**”. Complete the code and name your C code as **Array.c**. Have an output capture named **Array\_test.pdf** – containing at least 3 different runs.

So, your zip file will contain the following:

Two C files: **Histogram.c** and **Array.c** and two pdf files: **Histogram\_test.pdf** and **Array\_test.pdf**. Name your zip file with your last name first letter of your first name HW1.zip (ex: YasminSHW1.zip)

**Submission deadline is: 11:59 pm, Wednesday, February 10, 2021. No late submission will be considered.**