Programming Assignment 2

Fundamentals of Arrays

(Deadline: Oct 2, 23:59)

Instructions for the output: The below instructions apply to all the programs (even for the upcoming labs).

While printing the output, there is no new line

1. between any two output lines
2. after the last output line
3. Before the first input line

Please look at the end of this document (Appendix 1) for details in case you didn't understand the above instructions.

If the program fails the sanity check, it should just print **"error",** nothing less or more. Please note that "error" is different from "ERROR" and "Error" in C.

REGULAR EXERCISES

There are 3 regular exercises. You need to make only one program for all the 3 exercises. Since the exercises differ in the number of input parameters, the first argument to the program should be the exercise number. Refer to the test cases for further clarification.

Sanity checks required:

1. The number of input parameters should be in accordance with the exercise number.
2. You can assume the input parameters to be integers. However, you should verify them for their sign (positive/negative).

Exercise 1. Given the input, a 1-D array's lower bound (lb), upper bound(ub), base address(ba), words size(ws) and memory location(loc), output the following parameters.

1. Number of elements in the array
2. Total memory consumed to store the numbers
3. Memory location of a[i] (lb<=i<=ub)

Input format: ./a.out [exercise number] [lower bound] [upper bound] [base address] [word size] [location]

Output format: The output comprises of 3 lines as shown below

Number of elements

Memory consumed

Memory location of A[i]

Test Case1: ./a.out 1 -15 64 459 2 10 *(First argument is the exercise number)*

```
80
160
 509
```

Note: Please note the minus sign in the lower bound.

Test Case2: ./a.out 1 39 200 100 4 80

```
162
648
264
```

Test Case 3: ./a.out 1 10 20 2140 2 5 4

```
error
```

Test Case 4: ./a.out 10 20 2140 2 5 4

```
error
```

Exercise 2: Find the address of location A[I][J] of a 2-D matrix which is stored in a matrix by row major order. Following values are taken as inputs from the user.

(a) Number of rows (R)

(b) Number of column (C)

(c) Base address (BA)

(d) Word Size (W)

(e) I

(f) J

Input format: ./a.out [R] [C] [BA] [W] [I] [J]

Output format: Address Location

Test Case 1: ./a.out 2 10 20 2140 2 5 4 *(The first argument is the question number)*

```
2348
```

Test Case 2: ./a.out 2 20 30 1000 2 7 4

1428

Exercise3: Write a C program to find the address of location A[I][J] in a 2-D matrix which is stored in a matrix by column major order. Following values are taken as input from the user.

(a) Number of rows (R)

(b) Number of column (C)

(c) Base address (BA)

(d) Word Size (W)

(e) I

(f) J

Input format: ./a.out [R] [C] [BA] [W] [I] [J]

Output format: Address Location

Test Case 1: ./a.out 3 10 20 2140 2 5 4 *(The first argument is the question number)*

2230

Test Case 2: ./a.out 3 20 30 1000 2 7 4

1174

===

BONUS EXERCISES

There are 2 bonus exercises. Create separate programs for them.

Sanity checks required:.

1. You can assume the input parameters to be integers. However, you should verify them for their sign (positive/negative).

Exercise 1: Given two sets (in the form of 1-D arrays), say A and B. Print A∪B and A∩B, where ∪ and ∩ denote the union and intersection operations on sets.

Make sure that your logic for computing union and intersection processes the input sets linearly, first A and then B, (Refer to the test cases for further clarification).

Input format: ./a.out [number of elements in the first set] [elements of the first set separated by spaces] [number of elements in the second set] [elements of the second set separated by spaces]

Output format: The output comprises of 2 lines as shown below

[elements in A∪B separated by spaces]

[elements in A∩B separated by spaces]

Note: Ensure that the elements in one line of the output are separated by space and not some other whitespace character like tab or double spaces etc.

Sanity check required:

1. Since the input arrays are sets, ensure that there is no duplicate entry in the set (*since we are talking about sets, not multisets*)

Test Case 1: ./a.out 3 10 20 21 4 10 20 5 4

```
10 20 21 5 4
10 20
```

Test Case 2: ./a.out 2 3 4 1 2 6

```
error
```

Exercise 2: Given a 1-D array, write a program to find the kth maximum element in this array. The input to the program will be

1. Size of array
2. Array elements
3. k

Input format: ./a.out [number of elements in the array] [elements separated by spaces] [k]

Output format: The output is just one number (kth largest element)

Test Case 1: .a/.out 10 4 6 1 3 8 9 23 46 90 100 2

```
90
```

Test Case 2: .a/.out 10 4 6 1 3 8 9 23 46 90 100 5

```
9
```

Test Case 3: .a/.out 10 4 6 1 3 8 9 23 46 90 100 -1

```
error
```

Test Case 4: .a/.out 2 3 4 5

```
error
```

(Go to the next page for submission instructions)

SUBMISSION INSTRUCTIONS

Create separate folders for the submission of regular problems and the bonus problems. Name the folders as [your_roll_number]_[your_name]_regular and [your_roll_number]_[your_name]_bonus, respectively.

[your_roll_number]_[your_name]_regular: This folder should have one file named [your_roll_number]_[your_name]_p1.c.

[your_roll_number]_[your_name]_bonus: This folder should have two files named [your_roll_number]_[your_name]_p2.c (exercise 1) and [your_roll_number]_[your_name]_p3.c (exercise 2).

Zip both the folders together and name the zip file as [your_roll_number]_[your_name]_lab2. Upload the zip file to the Google classroom. Kindly use your complete roll number as assigned by the institute.

==

Appendix 1

Your program should not print any unnecessary new lines (or even tabs and spaces). As an example, assume that your code is printing 2 numbers as output, n (having value 10) and m (having value 40). Then your sequence of printf statements should be

=====================================
```
printf("%d",n);
printf("\n");
printf("%d",m);
```
This will result in an output like
```
10
40
```
=====================================

and not any of the following

1.
```
printf("%d",n);
printf("\n\n");
printf("%d",m);
```
Here, there is an additional newline between two output lines.

This will result in an output like
```
10

40
```
=====================================

2.

```
printf("%d",n);
printf("\n");
printf("%d",m);
printf("\n");
```

Here, there is an additional newline after the last output line.
This will result in an output like

```
10
40

```

=====================================

3.

```
printf("\n");
printf("%d",n);
printf("\n");
printf("%d",m);
```

Here, there is an additional newline before the first output line.
This will result in an output like

```

10
40
```

======================================