COP 3223 Assignment #7

Program A: Voting Machine (voting.c):

Voting with slips of paper is so archaic! So lets write a program that will manage voting for us. But lets take it one step further: lets make sure that each voter only votes once.

The program should input a number, N, the number of IDs that are allowed to vote. IDs in this program are the numbers 0 through N-1. The program will then prompt for votes from users. In order to place a vote, a user must enter first their ID number and then their vote. The possible votes are "A", "B", "C", and "D". If an ID number that has already voted is entered, then they should not be allowed to vote. If ID number -1 is entered, then the voting is over and the program should end, outputting the winner. You may assume that all inputs will be within the bounds specified above; you will not be required to check for invalid IDs, votes, etc. You may also assume that a tie will never occur at the end of all votes being added.

You may write any functions you find useful in this program, but you are not required to have any for this program.

Sample Run (user input in bold and italics):

```
4
What is your ID?
1
Welcome 1, which vote would you like to place?
A
You have successfully voted for A.
What is your ID?
2
Welcome 2, which vote would you like to place?
B
You have successfully voted for B.
What is your ID?
1
You have already voted. You cannot vote again.
What is your ID?
0
Welcome 0, which vote would you like to place?
A
You have successfully voted for A.
What is your ID?
-1
A wins with 2 votes!
```

<u>**Program B: Manual Sorting (manualsorting.c):**</u>

Evan claims to know how to sort lists of integers using only swap operations. You don't believe him, so you want to write a program to test his claim.

First, read in an integer, N, the number of numbers in the list. Then read in N numbers, the numbers in the list in its initial order. Following this will be a list of pairs of numbers. Each pair of numbers will be distinct from the other in the pair, and will be between 0 to N-1 (inclusive). These are Evan's swap indices. Apply each of these swaps to the array, in the order they are given. A swap of two indices of an array means that the values at each index are swapped such that they are now each at the opposite location. If at any time the array becomes sorted (from smallest to largest), say so. If the array ever becomes unsorted after being sorted, indicate this as well. The final line will contain two integers: -1 -1. This indicates Evan thinks he is done sorting the array. Output a message reporting if he was successful or not.

Your program should have and use the following function (written by you):

```
// Precondition: array will be an array of integers of length length.
// Postcondition: returns true if the array is in sorted
//    (nondecreasing) order, or false otherwise.
int is_sorted(int array[], int length);
```

It is also recommended that you use the `swap` function discussed in class.

Hint: this problem would be a really good place to use the print_int_array function suggested in class for debugging, since you can't see what's happening with the array using the required output. Even if your output is correct, it is a good idea to verify your program is actually doing what you expect it to. Just don't forget to remove that extra output before you submit.

<u>**Sample Run (User input in bold and italics):**</u>

```
3 2 1 3
What is the next swap?
1 2
What is the next swap?
1 2
What is the next swap?
0 1
Evan has sorted the array.
What is the next swap?
0 1
Evan has unsorted the array.
What is the next swap?
-1 -1
Evan has failed!
```

Sample Run (User input in bold and italics):

```
4
1 1 1 2
What is the next swap?
2 3
Evan has unsorted the array.
What is the next swap?
2 0
What is the next swap?
0 3
Evan has sorted the array.
What is the next swap?
-1 -1
Evan was right!
```

Program C: Range Minimum (rangemin.c):

We want to write a program to make it easy for us to keep track of the smallest value in a range of indices (locations) in a list. We also want to be able to update ranges within the list. Write a program that inputs a series of these actions.

Your program should first create an array of size N, where N is an integer read from the user. This array should start off with values all set to 0. Your program should then repeat the following process. First, read in a character from the user. If this character is an 'E', then end the program. If this character is an 'A', then it should read 3 integers to follow, L, H, and V. It should then increase the value of all locations in the array from L to H (L <= H) by V. If the character was a 'M', it will be followed by only two integers, L and H. Your program should in this case output the minimum value currently in the array between indices L and H (inclusive).

You may assume that all input values are within their specified bounds.

Your program should have and use the following functions (written by you):

```
// Precondition: low <= high. low >= 0. high < size of array.
// Postcondition: the value in each index of array between low and
//   high (inclusive) will be increased by amount.
void increment_range(int array[], int low, int high, int amount);

// Precondition: low <= high. low >= 0. high < size of array.
// Postcondition: returns the smallest value in the array between
//   index low and high (inclusive).
int get_range_min(int array[], int low, int high);
```

Hint: this is another program where you may find it very useful to have a function that prints out your array for debugging, since you can't directly see it. As usual, though, make sure there's no debugging output left over when you submit.

Sample Run (User input in bold and italics):

```
4
A 0 3 10
M 0 3
The minimum value currently in the range [0,3] is 10.
A 0 0 5
M 0 3
The minimum value currently in the range [0,3] is 10.
A 0 1 -6
A 3 3 -2
M 1 3
The minimum value currently in the range [1,3] is 4.
M 2 3
The minimum value currently in the range [2,3] is 8.
M 0 0
The minimum value currently in the range [0,0] is 9.
E
```

Deliverables:

Please submit three separate .c files for your solutions to these problems via WebCourses by the designated due date:

Program A: **voting.c**
Program B: **manualsorting.c**
Program C: **rangemin.c**