COP 3223 Recitation: Functions with Arrays

Problem A: Schedule Checker

You are very meticulous with your scheduling. Whenever you have a potential activity, you like to schedule it on the day with the fewest activities. You have already kept track of the number of activities for each day in the rest of the semester. Write a function that takes in this list, as well as a starting and ending range in days, and returns the day number with the fewest activities in that range. For example, if the input array looked like this:

index	0	1	2	3	4	5	6	7	8	9	10	11
# activities	8	3	6	7	9	5	3	8	6	7	4	5

and the start day of the request was 2 and the end day of the request was 9, the answer would be 6, since on day 6 there are only 3 activities and this is less than any of the other days.

Since you like getting things out of the way, if possible, if there are multiple days that are the "least busy" in the given interval, return the day that is earliest. (So for example, if you were given the query day 0 to day 10 with the array above, your function should return 1, since day 1 is the first day with only 3 activities planned.)

Write a function that takes in an array, a low index, a high index, and returns the index within low and high inclusive that stores the smallest number amongst array[low], array[low+1], …, array[high]. If more than one such index exists, return the lowest possible one.

```
// Pre-condition: low <= high and are both valid indexes to data.
// Post-condition: Returns the lowest index in [low, high] storing
//                 the minimum of array[low]…array[high].
int findMinIndex(int data[], int low, int high);
```

Problem B: Roommate Trouble!

You and your roommate were getting along, but just as you thought everything was fine, she started inviting random friends over to the room. In general, you have no problem with a few friends, but at some point it becomes too much. Luckily, by this point in the semester, you've learned how to deal with your problems maturely. You brought up a polite conversation with your roommate and she agreed to never bring more than 10 friends in any consecutive 7 days.

To make sure she's keeping up with her end of the bargain, you start recording how many friends she brings each day. You've decided, however, that it's too pain-staking to look through each 7 day interval to see if your roommate has violated the policy the two of you agreed upon. Thus, you've decided to write a computer program to make the calculation for you. Luckily, your gracious, affable C programming teacher has written some partial code to help you (since she commiserates with your roommate woes). All you have to do is write the function specified in the prototype below. Since your tastes may change, you've decided to write the function so that it can deal with any maximum number of friends for any consecutive period.

```
// Pre-condition: data has length elements. friendLimit, numDays are both
//                positive and numDays <= length;
// Post-condition: Returns 1 if there is some consecutive streak of numDays
//                 integers that adds up to more than friendLimit. Returns 0
//                 otherwise.
int tooManyFriends(int data[], int length, int friendLimit, int numDays);
```