CS 367 - Homework One

due Tuesday, 9/6 by 11:59pm

C refresher

The purpose of this homework assignment is to focuson how values are stored and manipulated in memory. The C language tends to keep us closer to the metal than other languages such as Java, and it will serve as our starting point as we dive deeper under the hood throughout the semester.

Task

You will write a simple program that interacts with standard input (*not* reading from a file), reads in integers for the variables k and for n, followed by n more integers (which should be placed in an array). The numbers will be separated only by some amount of whitespace. Then you will calculate the *kth-lowest* number in the array. This is the same as if you sorted the array and took the k-indexed value from a *one-indexed* array, but you are not expected to actually sort the array.

Your program will print the appropriate message:

- **bad index** (when `k` is non-positve or too large)
- **bad size** (when `n` is non-positive)
- the actual numeric answer should be printed when it exists

Hints

- Use of `scanf` is recommended to grab these integers.
- If you can calculate the answer using another array and a single traversal of the original array of numbers you read in, you will get extra credit (5%). We will be looking for the single loop and no recursion.
- don't forget the required bounds on `k` and on `n`: $1 \le k \le n$.
- As always, think ahead and break down the task into simpler steps. Think what storage you want to use, and the order in which you can create/will need to use different values you'll calculate along the way.
- If you want to sort the data, you must do so yourself for the practice; don't import and use a sorting routine! You are welcome to look up a search algorithm online and implement it, just cite where you went for guidance. You may not copy sorting code from online.
- Do more than a cursory bit of testing; there are plenty of orderings and corner cases to consider!

Rules

- you may not use the array creation syntax - that means you must use `malloc` instead of `int[] xs = ...`.

Examples

- `3 6 5 10 15 20 25 30`

```
15
```

- `2 4 3 9 8 6`

```
6
```

- `3 4 3 9 8 6`

```
8
```

- `5 6 2 4 2 3 4 5`

```
4
```

- `1 5 1 2 3 4 5`

```
1
```

- `-3 5 1 2 3 4 5`

```
bad index
```

- `100 3 1 2 3`

```
bad index
```

- `3 -5 1 2 3 4 5`

```
bad size
```

Submission

You will create a folder named `netID_h1` (use your own mason *netID*), and place two files in it:

- `h1.c`
- `Makefile`

Running the `make` command must create the executable named `homework1`.

You must compress this folder as a `.zip` file, and then upload it to Blackboard under the "homework" menu.

Rubric

Category	Percent	Notes
Submission Guidelines	5%	naming, folder structure, make behavior, etc.
Commenting	10%	meaningful, ample
test cases	85%	equal credit per test case used

Extraneous Penalties

- If you break rules such as using arrays instead of `malloc`, you will lose extra points. It will likely be around 5-15% for each, depending on how much of the exercise it circumvents. If you ended up not needing to write much code yourself because you imported something against the rules, then you will be awarded not many points as a result.