Spring, 2019 Name: _____

(Please *don't* write your id number!)

Practice Exam 3: C Programming with Conditionals and Loops

Directions

This exam is closed book and notes.
If you need more space, use the back of a page. Note when you do that on the front.
Before you begin, please take a moment to look over the entire test so that you can budget your time.
Clarity is important; if your answers are sloppy and hard to read, you may lose some points.

For Grading

Question:	1	2	3	4	Total
Points:	15	25	30	30	100
Score:					

1. (15 points) [Programming] In C, define the function specified in the following header file:

```
// $Id: cleararray.h,v 1.1 2017/04/04 19:40:51 leavens Exp $
#ifndef CLEARARRAY_H
#define CLEARARRAY_H 1
// requires: arr is allocated and has at least sz elements.
// modifies: arr[*], i.e., the elements of arr;
// ensures: for all i such that 0 <= i and i < sz, arr[i] == 0.0.
extern void cleararray(double arr[], int sz);
#endif
```

that takes an array of **double**s, `arr`, of size `sz`, and assigns to each element of `arr` the **double** zero (i.e., `0.0`). You many not use any C library functions in your solution. Tests for this problem appear below.

```
#include <stdbool.h>
#include "tap.h"
#include "cleararray.h"
// ensures: result is true just when all the elements in a are equal to 0.0.
static bool checkIt(const double a[], const int sz) {  // A testing helper,
    for (int i = 0; i < sz; i++) {                     // not for you to write!
        if (a[i] != 0.0) { return false; }
    }
    return true;
}
int main() {
    double a5[5] = {0.5, 1.5, 2.5, 3.5, 4.5};
    cleararray(a5, 5);
    ok(checkIt(a5, 5), "checkIt(a5, 5)");
    double a10[10] = {0.01, 1.14, 2.73, 3.14, 4.44, 5.97, 6.22, 7.31, 8.99, 9.99};
    cleararray(a10, 10);
    ok(checkIt(a10, 10), "checkIt(a10, 10)");
    double a2000[2000];
    cleararray(a2000, 2000);
    ok(checkIt(a2000, 2000), "checkIt(a2000, 2000)");
    return exit_status();
}
```

2. (25 points) [Programming] In C, define the function specified in the following header file:

```
// $Id: countelems.h,v 1.2 2017/04/05 01:22:22 leavens Exp leavens $
#ifndef COUNTELEMS_H
#define COUNTELEMS_H 1
// requires: arr is allocated and has sz elements
// ensures: result is the number of times sought appears as an element in arr
extern int countelems(const int sought, const int arr[], const int sz);
#endif
```

that when given an int, `sought`, and an array of ints, `arr` of size `sz` elements returns the number of times that `sought` occurs in the array `arr`. Tests for this problem appear below.

```
#include "tap.h"
#include "countelems.h"
int main() {
    int a1[1] = {32};
    ok(countelems(32, a1, 1) == 1);
    ok(countelems(9, a1, 1) == 0);
    int a2[2] = {32, 23};
    ok(countelems(32, a2, 2) == 1);
    ok(countelems(23, a2, 2) == 1);
    ok(countelems(10, a2, 2) == 0);
    int a5[5] = {0, 5, 10, 5, 0};
    ok(countelems(0, a5, 5) == 2);
    ok(countelems(10, a5, 5) == 1);
    ok(countelems(5, a5, 5) == 2);
    ok(countelems(15, a5, 5) == 0);
    int a300[300]; // fill with 0, -1, -2, -3, -4, -5, 0, -1, ...
    for (int i = 0; i < 300; i++) { a300[i] = -(i % 6); }
    ok(countelems(0, a300, 300) == 50);
    ok(countelems(-1, a300, 300) == 50);
    ok(countelems(-2, a300, 300) == 50);
    ok(countelems(-3, a300, 300) == 50);
    ok(countelems(-4, a300, 300) == 50);
    ok(countelems(-5, a300, 300) == 50);
    ok(countelems(-6, a300, 300) == 0);
    ok(countelems(23, a300, 300) == 0);
    return exit_status();
}
```

3. (30 points) [Programming] In C, define the function specified in the following header file:

```
// $Id: first_dup.h,v 1.2 2017/04/05 00:57:53 leavens Exp $
#ifndef FIRST_DUP_H
#define FIRST_DUP_H 1
// requires: str is a null-terminated string that is allocated and assigned
// ensures: result is the first char in str that is duplicated in str, if any,
//          and '\0' if there are no duplicate characters in str.
extern char first_dup(const char *str);
#endif
```

that takes a string, `str`, and returns the first character in `str` that is duplicated within `str`, or `'\0'` if no characters are duplicated in `str`. A character c is *the first duplicate* if, for every other character x in `str` that occurs more than once, the first occurrence of c in `str` is to the left of the first occurrence of x (i.e., c occurs at a lower index than x). Your solution can use C library functions, if you include the appropriate header file. Examples appear below.

```
#include "tap.h"
#include "first_dup.h"
int main() {
    ok(first_dup("") == '\0');
    ok(first_dup("a") == '\0');
    ok(first_dup("aa") == 'a');
    ok(first_dup("bab") == 'b');
    ok(first_dup("aah") == 'a');
    ok(first_dup("caaah") == 'a');
    ok(first_dup("Mississippi") == 'i');
    ok(first_dup("The bookshelf is brimming") == 'h');
    ok(first_dup("Hah, case matters!") == 'a');
    ok(first_dup("The quick brown fox quivvered.") == 'e');
    ok(first_dup("abcdefghijklmnopqrstuvwxyzm") == 'm');
    return exit_status();
}
```

4. (30 points) [Programming] In C, define a program that prompts, on standard output, with "`how many? `" and then reads a positive integer from standard intput, n. The number n is guaranteed to be no larger than 1000. It then reads n doubles, which are all between 0.0 and 10^5 from standard input and prints out both the minimum and the maximum of the numbers read in the format:

```
minimum is: 3.75
maximum is: 7.514
```

Sample interactions appear below, where the text after the "`? `" in the prompt, and the n lines that follow are all typed as standard input. You may assume that the first number n is strictly positive (i.e., it is at least 1), less than 1000, and that all of the doubles read are non-negative and no greater than 10^5.

A first interaction:

```
how many? 3
2.0
4.0
6.0
minimum is: 2.000000
maximum is: 6.000000
```

A second interaction:

```
how many? 6
3.751429
3.75
3.75
7.514
6.32941
5.3756
minimum is: 3.750000
maximum is: 7.514000
```

A third interaction:

```
how many? 1
10.0e5
minimum is: 1000000.000000
maximum is: 1000000.000000
```

Hint, use the format specifier `%lf` to read a double with `scanf`. Use the format specifier `%f` to print a double in the specified format with `printf`.

```
//more space for your solution
```