CS 24000: Programming in C
Midterm Exam 1
Fall 2018

Name: □

Username: □□□□□□□□

Read all instructions before beginning the exam.

- This is a closed book examination. No material other than those provided for you are allowed.
- You need only a pencil and eraser for this examination. If you use ink, use either black or blue ink. If you use pencil, your writing must be dark and clearly visible.
- This examination contains an amount of material that a well-prepared student should be able to complete in well under one hour.
- This examination is worth a total of 100 points. Not all questions are worth the same amount. Plan your time accordingly.
- Write legibly. You should try to adhere to the course code standard when writing your solution(s). Egregious violations may result in point deductions.
- You may leave after you have turned in all pages of the examination booklet. You will not be able to change any answers after turning in your examination booklet.
- Read each question *carefully* and *only do what is specifically asked for* in that problem.
- Some problems require several steps. Show all your work. Partial credit can only be rewarded to work shown.
- Do not attempt to look at other students' work. Keep your answers to yourself. Any violation will be considered academic dishonesty.
- Write your username on *EVERY* page where indicated. Any page without a username will receive a zero for the material on that page.
- Read and sign the statement below. Wait for instructions to start the examination before continuing to the next page.

"I signify that the answers provided for this examination are my own and that I have not received any assistance from other students nor given any assistance to other students."

Signature:

- Do not open the examination booklet until instructed.

1. (20 points) Write short answers to the following questions.

   (a) (2 points) Write a single, valid command with which you would use gcc to compile a C file named `abc.c` into an object file called `xyz.o` with warnings treated as errors and including support for debugging. There are multiple valid answers.

   (b) (2 points) Write a single, valid command with which you would link three files named `xyz.o`, `abc.o`, and `def.c` together into an executable named `prog`. There are multiple possibilities.

   (c) (2 points) Describe briefly what the `-Wall` flag does when passed to gcc.

   (d) (2 points) What do many of the functions found in the string library (with prototypes in `string.h`) rely on to operate correctly?

   (e) (2 points) What allocates memory for a variable—a declaration or a definition?

   Given the following code segment and a 64-bit architecture, answer questions 1.f. through 1.i.

```
int array[] = { 12, 5, 3, 6, 9, 2, 4, 2} };
int *ptr = 0;
ptr = &(array[2]);
printf("size = %d\n", sizeof(array);
```

   (f) (2 points) Assuming that `sizeof(int) = 4`, what was displayed?

   (g) (2 points) What is the value of `*(ptr - 1)`?

   (h) (4 points) What is the value of `*(ptr - *(ptr + *ptr))`?

   (i) (2 points) What is the value of `sizeof(ptr)`?

2. (20 points) Answer the following questions.

   (a) (4 points) Declare a structure, `resistor_struct`, which contains (in this order) an array of `ID_LEN` (=5) characters named `id`, a float named `max_power`, and an integer named `resistance`.

   (b) (2 points) Use `typedef` to declare a new type, `resistor_t`, of the structure in 2.a.

   (c) (4 points) Declare a structure, `circuit_struct`, that contains an array of `CNAME_LEN` (=24) characters named `name` and an array of 10 `resistor_t`'s, called `resistors`.

   Given the following: `sizeof(int) = 4`, `sizeof(float) = 4`, `sizeof(char) = 1`, answer 2.d. and 2.e.

```
struct circuit_struct circuit_board[5];
printf("sizeof = %d\n", sizeof(circuit_board);
```

   (d) (4 points) What would be printed to the screen?

   (e) (6 points) Write a function called `find_voltage()` that accepts two parameters—one of type `resistor_t` and one of type `int`. The function should return an integer representing the voltage, calculated by multiplying the integer argument (the current) by the `resistance` member of the first argument.

3. (40 points) Write a function, `blown_resistors()`, that returns an integer (the number of resistors that have exceeded their `max_power` value), and writes the `id` of each blown resistor as well as the power it is dissipating to a file. The parameters to the function are the input filename, the output filename, and the net voltage across the entire circuit (a float).

The input file will describe a single circuit. Each record in the input file is comma delimited and describes a resistor present in the circuit. Each line (terminated by a newline character) contains the `id` (string shorter than `ID_LEN` length), `resistance` (integer) in ohms, and maximum power—`max_power` (float). Use the structure that you declared in part 2.a. of this exam to hold these values.

In order to determine if a resistor is blown, you must calculate the actual power that is being dissipated. This is determined by the following equation:

$$power = current^2 * resistance \tag{1}$$

If the power being dissipated is greater than the resistor's `max_power` threshold, the resistor is consider to have "blown."

`current` is calculated using the following equation and *is the same for every resistor*:

$$current = net\_voltage/net\_resistance \tag{2}$$

You must first calculate the total resistance of the entire circuit—`net_resistance` (found by summing all of the resistances). Once determined, you will be able to calculate the `current` flowing through all resistors using the above equation (2).

The output file should contain the `id` of each blown resistor followed by a comma, followed by a space, followed by the actual power being dissipated to two decimal places, followed by a newline character.

DO NOT assume that the data in the input file is without error (HINT: each record must have three fields). Also, ensure that the files have been properly opened. If ANY error occurs, return -1. Otherwise the function should return the number of blown resistors (int). Do not forget to set the file pointers back to `NULL`.

FINAL HINT: Use `fseek()` to jump back to the beginning of the file and read the data twice. The first time through, calculate the `net_resistance`; the second time through, determine which resistors have blown.

Here is an example of an input file and the corresponding output file generated, assuming the net voltage is 100.00:

```
input file:                     output file:
R A, 5, 25.50                   R B, 80.00
R B, 20, 75.00                  R D, 40.00
R C, 15, 80.25
R D, 10, 5.15
```

The return value would be: 2

Use the following sheet to write your code. Follow the code standard as much as possible, but do not spend too much time on comments.

Work area for problem 3...

Turn over for additional space...

Work area for problem 3 continued...

4. (20 points) Given the following structure declaration:

```
struct coord {
  float x;
  float y;
};
```

Write a function, `find_center()`, that accepts a parameter called `file_ptr` that is a `FILE` pointer for a file that has already been opened for binary read, and returns a `struct coord`. The open file contains an unknown number of binary-format `struct coord`'s that must be read. The function should calculate the average `x` and `y` values of all coordinates in the file. When done, the function should return a `struct coord` with its `x` and `y` values set to their respective average. Your function should not close the binary file. Assume no errors occur.

Turn over for additional space...

Work area for problem 4...