CS 240: Programming in C
Midterm Exam 1
Fall 2024

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. Function headers are not required.
- 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.
- If you finish your exam with less than 10 minutes remaining, please remain seated until the exam is over. We will call you down by row to turn in your exam.
- 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. (25 points) Write short answers to the following questions.

   (a) (2 points) Given the C source file `player.c` and the compiled object files `game.o` and `level.o`, write two GCC commands to do the following:

   1. Compile `player.c` with debug symbols, all warnings enabled, and all warnings treated as errors.
   2. Link all of the objects together into an executable named `game`.

   Assume all files are in the current working directory.

   (b) (2 points) Describe briefly what the `-I` compiler flag does and why one would use it.

   (c) (2 points) What is a function prototype, and when is it necessary to use one? What parts of the function does it describe?

Consider the following array of characters, where `'\0'` represents the NUL terminator. Then answer parts (d)-(f).

```
char str[] = {'H','e','l','l','o','\0','W','o','r','l','d','\0'};
```

(d) (1 point) What will be the return value of `strlen(str)`?

(e) (1 point) What will be the return value of `sizeof(str)`?

(f) (1 point) What will be printed to stdout by `printf("%s", str)`?

(g) (2 points) Consider the following code to read in a student's username and score from an open file pointer `fp`.

```
char username[NAME_LENGTH];
int score;
int ret = fscanf(fp, "%s %d", username, &score);
```

Briefly explain why `fscanf` requires the ampersand (&) to read in an integer value. Why doesn't it require one to read in a string?

(h) (4 points) Given the following call to `scanf`:

```
int ret = scanf("%d, %s, %d, %s", &num1, str1, &num2, str2);
```

What will be the value of `ret` if stdin contains the following line?

```
6, abc, 12, def
```

Will this format string correctly read 6, "abc", 12, and "def" into `num1`, `str1`, `num2`, and `str2`, respectively? If not, why not? How would you change the format string to do so?

(i) (5 points) Given the following structure, indicate the location of each variable in memory by labeling the bytes in the diagram below with the letter of the variable occupying that space. Each box represents one byte. Use P for padding, and leave blank any bytes that are not part of the structure. Assume a 64-bit architecture.

```
struct a_struct {
  short S;
  char C[2];
  long L;
  int I;
} my_struct;
```

Consider the following variable, and answer parts (j) and (k).

```
int flags = 45; /* binary 00101101 */
```

(j) (2 points) Using only bitwise operators (&, |, ~, <<, >>) and assignment (=), write a single line of code to set bit 6 (i.e., the value of bit 6 should be changed to 1). Note: the right-most bit is bit 0.

(k) (3 points) Write a single line of code to clear bits 2 through 4, using only bitwise operators, assignment, and optionally subtraction.

2. (20 points) Answer the following questions.

   (a) (4 points) Declare a structure named `star_struct` which contains (in this order) an array of `STAR_NAME_LEN` characters named `name`, an array of `CONS_NAME_LEN` characters named `cons_name`, a float named `vis_mag`, and a float named `parallax`.

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

   (c) (4 points) Declare a structure named `cons_struct` which contains an array of `CONS_NAME_LEN` characters named `name`, a float `total_mag`, an array of `STAR_NAME_LEN` characters named `brightest_star`, and a float `brightest_star_mag`. In the same statement, use `typedef` to declare a new type, `cons_t`, of the structure.

   (d) (2 points) Say you have the following definitions:

```
#define STAR_NAME_LEN (18)
#define CONS_NAME_LEN (26)
```

   Given `sizeof(float) == 4` and `sizeof(char) == 1`, what does the following statement print?

```
printf("%ld\n%ld\n", sizeof(star_t), sizeof(cons_t));
```

(e) (4 points) A star's brightness as viewed from Earth is referred to by astronomers as its *apparent magnitude.* This unitless quantity behaves on a reverse logarithmic scale – the **brighter** a star is, the **lower** it's apparent magnitude. We can also describe a star's intrinsic luminosity, known as the *absolute magnitude.* If we're given the star's distance d from Earth in parsecs along with its apparent magnitude m, we can calculate the absolute magnitude M as such:

$$M = m - 5(\log_{10} d - 1). \tag{1}$$

The distance to a star can be roughly estimated by observing its *stellar parallax* – the apparent shift in position when viewed from different orbital positions of Earth. Given the parallax p in arcseconds, we can obtain the distance of a star in parsecs as such:

$$d = \frac{1}{p}. \tag{2}$$

Write a function `calc_abs_mag` that calculates and returns the absolute magnitude of a star as a float. The first argument is a float `app_mag` with the apparent magnitude, and the second argument is a float `parallax` with the stellar parallax in milli-arcseconds.

Hint: you can use the `log10f()` function to calculate the base-10 logarithm of a float.

Hint: there are 1000 milli-arcseconds in one arcsecond.

(f) (4 points) Oftentimes we want to combine magnitudes to get the overall brightness of a group of stars. This can be done by converting to linear units, adding, and then converting back into the magnitude scale:

$$m_f = -2.5 \log_{10}(10^{-0.4 \times m_1} + 10^{-0.4 \times m_2}) \tag{3}$$

Write a function `combine_mags` to calculate and return the combined magnitude as a float. It takes two float arguments, `mag1` and `mag2`.

Hint: you can use the `powf(float base, float exponent)` function to calculate a quantity raised to an exponent.

3. (30 points) Write a function, `brightest_constellation()`, that determines the constellation with the highest combined intrinsic luminosity of all the stars that comprise it. The parameters to the function are the input filename (a string) and the output filename (a string). It should return an integer: `0` on success or `-1` on failure.

   The input file consists of a list of stars and some data about each, with one star per line. The format of each line is given below:

   `name|bayer_designation|ra_h|ra_m|dec|class|app_mag|parallax|error|notes`

   - `name` is the proper name of the star, which is at most `STAR_NAME_LEN - 1` characters long and may contain spaces.
   - `bayer_designation` is a naming scheme for stars that consists of a Greek or Latin letter, followed by a space, followed by the name of the star's parent constellation. You will use this to determine which constellation a star belongs to. Constellation names are at most `CONS_NAME_LEN - 1` characters long and may contain spaces.
   - `app_mag` is the apparent magnitude.
   - `parallax` is the stellar parallax in milli-arcseconds.

   The other fields are unimportant, so we can ignore them. (Hint: use `*` in your format specifier to skip over it, e.g. `"%*s"` will read a string but not store it.)

   A sample input file follows:

```
Mintaka|Delta Orionis|05|32|-0.3|09.5II+B2V|2.25|3.56|0.83|Right-most star system in Orion's belt
Bellatrix|Gamma Orionis|05|25|+6.3|B2III|1.64|13.42|0.98|Blue giant, ~7.7x sun's mass
Vega|Alpha Lyrae|18|37|+38.8|A0V|0.03|128.93|0.55|First star to be photographed
Polaris|Alpha Ursae Minoris|02|32|+89.3|F7Ib-II|1.99|7.56|0.48|Commonly called the North Star
Rigel|Beta Orionis|05|15|-8.2|B8Ia|0.15|4.22|0.81|Strong stellar winds
Regulus|Alpha Leonis|10|08|+12.0|B7V|1.36|42.09|0.79|Quadruple star system
Pollux|Beta Geminorum|07|45|+28.0|K0III|1.14|96.74|0.87|Closest red giant to the sun
Sirius|Alpha Canis Majoris|06|45|-16.7|A1V|-1.46|379.21|1.58|Brightest star in the night sky
Betelgeuse|Alpha Orionis|05|55|+7.4|M2Ib|0.55|7.63|1.64|Red giant, ~700x sun's radius
```

   The output file should contain the name of the constellation with the brightest combined absolute magnitude of all the stars within it, followed by the value of the combined absolute magnitude, and finally the name of the brightest star in that constellation. A sample output file corresponding to the above input file follows:

```
Orionis
-7.119197
Rigel
```

   Assume we have the following global definitions:

```
#include <stdio.h>
#include <string.h>
#include <math.h>

#define STAR_NAME_LEN (18)
#define CONS_NAME_LEN (26)
#define NUM_CONS      (88)

cons_t g_cons[NUM_CONS];
```

   You may also use the structs, types, and functions you created in question 2.

DO NOT assume that the data in the input file is without error. Also, ensure that the files have been properly opened. If ANY error occurs, return -1. Make sure to properly close any open files and set the file pointers back to `NULL`.

Hint: You must create each constellation the first time you encounter it while reading stars from the input file by filling in the global `g_cons` array. If the constellation already exists, combine the current star's absolute magnitude with the constellation's total absolute magnitude. Use `strcmp()` to compare constellation names.

Turn over for additional space...

Work area for question 3 continued...

Turn over for additional space...

Work area for question 3 continued...

4. (25 points) Answer the following questions.

(a) (5 points) In computer graphics, 3D geometry is typically represented by a collection of triangles called a mesh. Each triangle is composed of three vertices, with each vertex having an x, y, and z coordinate. Rotation of such a mesh is achieved by multiplying each vertex v with a 3 × 3 rotation matrix R, as shown:

$$\begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = \begin{bmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \\ R_{20} & R_{21} & R_{22} \end{bmatrix} \times \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \qquad (4)$$

or, equivalently,

$$\begin{aligned} v'_x &= R_{00} \times v_x + R_{01} \times v_y + R_{02} \times v_z \\ v'_y &= R_{10} \times v_x + R_{11} \times v_y + R_{12} \times v_z \\ v'_z &= R_{20} \times v_x + R_{21} \times v_y + R_{22} \times v_z. \end{aligned} \qquad (5)$$

Assume you have the following structure declaration:

```
struct vertex {
  float x;
  float y;
  float z;
};
```

Write a function named `rotate_vert` that takes a `struct vertex` argument and a `float[3][3]` rotation matrix argument which returns a rotated `struct vertex` using equation 5 above.

(b) (20 points) In addition to the `struct vertex` from part (a), assume you are given the following structure declaration for a triangle:

```
struct triangle {
  struct vertex v1;
  struct vertex v2;
  struct vertex v3;
};
```

Write a function named `rotate_mesh` that rotates a mesh of triangles stored within a binary file. The first argument is a `FILE *` pointer to a file that has been opened in read/write binary mode. The second argument is a `float[3][3]` rotation matrix. The return value is the total number of vertices that were rotated.

The binary file begins with an `int ntris` that counts the number of triangles in the mesh, followed by `ntris` × `struct triangle` triangles. You should read each triangle, rotate its vertices using your `rotate_vert` function from part (a), and then write the rotated triangle back to the file, replacing the original triangle.

DO NOT try to read all the triangles in at once. Assume there can be a very large number of them.

You should not open or close the file, and you may assume that no errors occur.

Turn over for additional space...

Work area for question 4 continued...