CS 240: Programming in C
Midterm Exam 1
Spring 2025

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 approximately two hours.
- 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.

Submission #: _____

1. (25 points) Write short answers to the following questions.

   (a) (2 points) Given the C source files `starship.c` and `sensors.c`, write two GCC commands to compile each into an object file. Include debug symbols, enable all warnings, and adhere to the C17 standard.

   (b) (2 points) Given the precompiled object file `warpdrive.o`, along with the output object files from part (a), write a single GCC command to link all object files into a single executable named `boldly_go`. Then, write a single command to execute the program.

   (c) (3 points) Given the code segment below:

```
char first[8];
int second;
double third;
int ret = 0;
ret = scanf(/* format string */, first, &second, &third);
```

   complete the code by writing a format string that completely matches each of the input lines below. In other words, `ret` must equal 3 after reading any of the below lines.

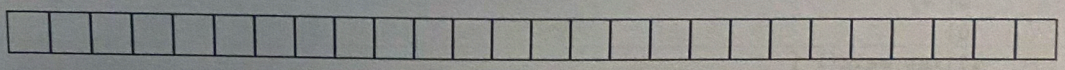
```
* abc  (29+3.5
  *0x7f(777+8.6
**_/:\_* (37+6
```

   (d) (2 points) What will be printed by this code segment?

```
char str[] = "ABCDEFG";
str[3] = '\0';
printf("%d %d", strlen(str), sizeof(str));
```

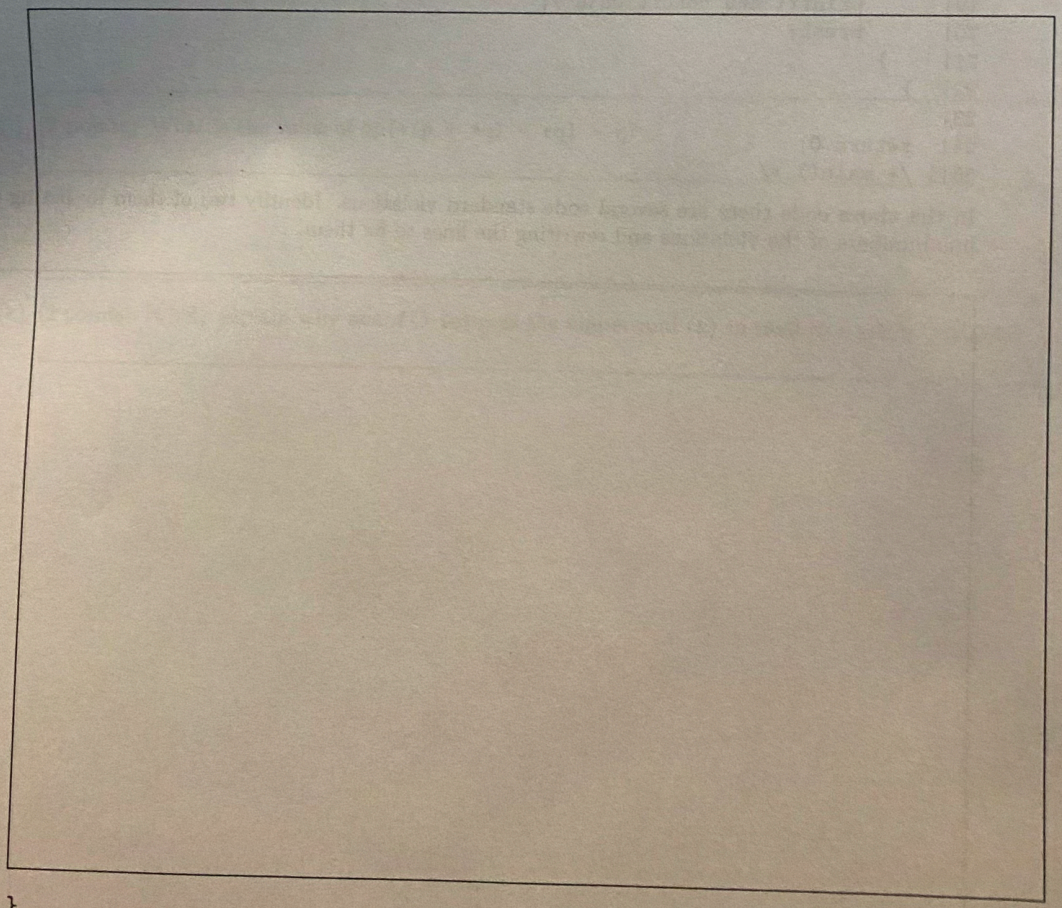
(e) (3 points) Given the following structure, indicate the location of each variable in memory by labeling the bytes in the diagram below with the letter of the variable occupying that space. Each box represents one byte. Use P for padding, and leave blank any bytes that are not part of the structure. Assume a 64-bit architecture.

```
struct a_struct {
  char z;
  short y;
  char x[2];
  double w;
  int v;
} my_struct;
```



(f) (3 points) Implement the function `set_clear()` below. The function should modify the bit at position `pos` in `number` according to the `mode`. If `mode` is equal to the constant `BIT_SET`, set the bit. If `mode` is equal to `BIT_CLEAR`, clear the bit. Otherwise, do not modify the bit. Return the number after applying the set or clear operation. You must use bitwise operators.
Remember: bit positions are 0-indexed starting from the right-most bit.

```
int set_clear(int number, int pos, int mode) {
```



```
}
```

(g) (2 points) Consider the following source file named `midterm1.c`:

```
 1|#include "midterm1.h"
 2|
 3|#include <stdio.h>
 4|
 5|/*
 6| * This function repeatedly reads two integers from stdin until a
 7| * specific condition is met, and then returns 0. If a read error
 8| * occurs, it returns -1.
 9| */
10|
11|int main() {
12|  while (1) {
13|    int a, b;
14|    int ret = scanf("%d %d", &a, &b);
15|
16|    if (ret != 2)
17|      return -1;
18|    if (a < b || a == b * b) {
19|      printf("End condition\n");
20|      break;
21|    }
22|  }
23|
24|  return 0;
25|} /* main() */
```

In the above code there are several code standard violations. Identify two of them by listing the line numbers of the violations and rewriting the lines to fix them.

(h) (2 points) Given the following code segment, what will be output by the `printf()` statement? Assume the code executes on a little-endian architecture.

```
union data {
  int num;
  char code[4];
} info = { .code = { 0xce, 0xfa, 0x1d, 0xb0 } };
printf("%x", info.num);
```

Given the following code segment, answer parts (i) and (j).

```
short array[] = { 1, 3, 5, 7, 11 };
short *p = array;
short *q = p + 1;
```

(i) (2 points) What is the value of `*(q + 2)`?

(j) (2 points) What is the value of `&p[*(p + *q) - *q] - q`?

(k) (2 points) Briefly explain why `scanf()` requires the ampersand (`&`) to read in a single variable.

(h) (2 points) Given the following code segment, what will be output by the `printf()` statement? Assume the code executes on a little-endian architecture.

```
union data {
  int num;
  char code[4];
} info = { .code = { 0xce, 0xfa, 0x1d, 0xb0 } };
printf("%x", info.num);
```

Given the following code segment, answer parts (i) and (j).

```
short array[] = { 1, 3, 5, 7, 11 };
short *p = array;
short *q = p + 1;
```

(i) (2 points) What is the value of `*(q + 2)`?

(j) (2 points) What is the value of `&p[*(p + *q) - *q] - q`?

(k) (2 points) Briefly explain why `scanf()` requires the ampersand (`&`) to read in a single variable.

2. (15 points) Answer the following questions.

(a) (2 points) Declare a structure, `coord_system`, which contains three `doubles`: `x`, `y`, and `z`.

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

(c) (3 points) Declare a structure, `box_struct`, that contains a string of `LABEL_LEN` (=18) elements named `label` and an array of 8 `coord_t`'s, called `corner`.

Given the following: `sizeof(double) = 8`, `sizeof(float) = 4`, `sizeof(char) = 1`, answer 2.d. and 2.e.

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

(d) (2 points) What would be printed to the screen?

(e) (6 points) Write a function called `scale_box()` that accepts two parameters—one of type `struct box_struct` and one of type `double`. The function should return a `struct box_struct` where each coordinate is the corresponding first argument's coordinate multiplied by the second argument.

3. (40 points) For people that enjoy kayaking, it is often desirable to know the condition of the body of water on which they'll be maneuvering. The United States Geological Survey (USGS) maintains a large number of monitoring stations. Unfortunately these gauges typically measure discharge in cubic feet per second (cfs), while kayakers are often interested in speed in miles per hour.

   Because discharge is measured as volume per unit time, one must first know the cross-sectional area of the river at the point of measurement before obtaining the flow velocity.

   River speed is given by the following equation:

$$V = R/A \tag{1}$$

   where V is river speed, R is discharge rate, and A is cross-sectional area.

   Write a function named `river_speed` that determines and returns the average speed (a `double`) in miles per hour for a given day and monitoring station ID.

   It should accept three arguments (in this order): the input filename (a string), the day (an integer), and the station ID (a string).

   The input file should be in the following format:

```
station_id|area
station_id|area
...
day:time:station_id:flow_rate
day:time:station_id:flow_rate
...
```

   `station_id` is a string containing only capital letters and digits 1-5 of at most `MAX_LEN - 1` length. Note that `MAX_LEN = 8`.

   `area` is a double precision floating point number indicating cross-sectional area for that station in square feet.

   `day` is an integer ranging from 1 to 365 representing the day of the year. Each file contains data for one year.

   `time` is an integer field containing seconds elapsed since midnight for that day.

   `flow_rate` is a double precision floating point number indicating the measured rate of water flow in cubic feet per second (cfs).

   A sample input file follows:

```
WAB1|1672.7
WAB2|1916.2
CO1|2136.88
CO2|1783.2245
CO3|1893.2555
17:3600:WAB1:36877.234
17:18032:CO1:287153.67898
18:10800:CO1:297563.22
20:3600:WAB1:35350.667
20:18032:WAB1:30340.432
20:82800:WAB1:28760.09806
195:10800:CO3:233000.23
```

Note that for the first set of data, fields are separated by a pipe (|). For the second set of data, a colon (:).

Note also that there are 5,280 feet in a mile.

DO NOT assume that data in the input file are without error. Strings that are not able to be stored in `MAX_LEN` (=8) bytes should be considered to be one of the possible errors. Also, ensure that the file has been properly opened. If ANY error occurs, return -1. Do not forget to set the file pointer back to `NULL`.

As an example, the return value for WAB1 on day 20 would be: 12.833252

Be sure to include appropriate assertion checks for the function arguments. You should provide the appropriate `#includes` as well.

Follow the code standard as much as possible, but do not spend too much time on comments.

Hint: Pay close attention to `fscanf()`'s return value. Use `ftell()` and `fseek()` to attempt to re-read a record when the format changes.

Do not write code on this page, it will not be graded.

4. (20 points) You are given a binary file containing a database of audio clips, and your task is to write a function to play a specific audio clip from the file. The database consists of two parts: an index of all the audio clips, followed by the raw audio data. The index begins with an `int num_clips`, followed by an array of `num_clips` number of `struct audio` structures. The `struct audio` is declared below:

```
struct audio {
  int id;
  int offset;
  int size;
};
```

The `offset` indicates the byte offset within the file where the raw audio data for this clip begins, and `size` indicates the number of bytes of raw audio that comprise the clip.

Write a function named `play_audio()` that accepts two arguments and returns an `int`. The first argument is the `FILE` pointer of the audio database that is already open in binary read mode. The second argument is an `int` specifying the ID of the audio clip to play. The function should search the index for the audio clip with the matching ID, and then play the clip. If the ID was not found in the index, return `NOT_FOUND`. Otherwise, return `SUCCESS`.

To play an audio clip, you must read the raw audio data into a buffer, and then call the `buffer_audio()` function, whose prototype is listed below.

```
void buffer_audio(char buffer[], int size);
```

However, `buffer_audio()` can only accept buffers up to `BUF_SIZE` bytes in length. Therefore, you must read the raw audio in chunks of `BUF_SIZE` bytes and repeatedly call `buffer_audio()` until you have read the entire clip.

Your function should assert that the first argument is non-`NULL` and that the second argument is non-negative. Do not close the file. You may assume that no read errors occur.

Below is a hex dump of an example audio clip database file containing two audio clips with IDs 5 and 14. The raw audio data starts at offset `0x1c`.

```
00000000: 0200 0000 0500 0000 1c00 0000 3b00 0000  ............;...
00000010: 0e00 0000 5700 0000 5800 0000 5e68 381f  ....W...X...^h8.
00000020: cba6 4284 69ce 6dc3 38d9 6351 f558 efcd  ..B.i.m.8.cQ.X..
00000030: d6f2 314d 01b1 08c7 ed12 bf06 93d4 2bf9  ..1M..........+.
00000040: 1d7b a44e ccac 2a75 8792 6f7a ec60 ef66  .{.N..*u..oz.`.f
00000050: 18e9 48bb 4e57 240e a82a 8bf7 577d 6e77  ..H.NW$..*..W}nw
00000060: c9b5 57c5 5ff3 9887 d510 499a 03ac ba80  ..W._.....I.....
00000070: d2e7 7ed7 b349 a350 359a 5ec4 51ce 9c16  ..~..I.P5.^.Q...
00000080: 0ccc 2b39 8c16 174f 0c3c 652c ea60 8f1d  ..+9...O.<e,.`..
00000090: 58ce 7d97 ae97 67fd 422b b807 cbca bbb3  X.}...g.B+......
000000a0: 85bb 2c5c a7dd 524c 56f4 2bce 82e3 c9    ..,\..RLV.+....
```

Write your answer on the next page.