CS 240: Programming in C
Midterm 1
Spring 2025

Practice Midterm 1

Version I

Name:

Username:

Read all instructions before beginning the exam.

- This exam is intended to be equally or more difficult than the past official midterm provided.
- You are encouraged to post on Ed Discussion, without hesitation, any sort of question you may have about this practice 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 less than 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.
- Read each question carefully and only do what is specifically asked for in that problem.
- Assume appropriate includes have been added to the code segments shown in the problems.
- Some problems require several steps. Show all your work. Partial credit can only be rewarded to work shown.
- Write your username on EVERY page where indicated. Any page without a username will receive a zero for the material on that page.

Signature:

Do not open the examination booklet until instructed.

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

   (a) (2 points) Given the files, `coords.c`, `chunks.o`, and `coords.h` write two one-line GCC commands to do the following:

   1. Compile `coords.c`, which includes `coords.h`, with all warnings enabled, disabled optimizations, and in adherence to the ANSI standard.

   2. Link all objects together into an executable named `terrain`.

   All files are in the current working directory.

   (b) (2 points) Briefly explain what is the purpose of the GCC flag, `-DNDEBUG=TRUE`.

(c) (2 points) What is the output of the following program? State the value displayed by `printf()`, undefined behavior, or error produced.

```
int main() {
    char a[] = "Turkstra";
    char *b = "May";
    strncpy(a, b, strlen(b));
    printf("Output: %s\n", a);
}
```

(d) (2 points) Rewrite the following expression using `fscanf()` in one line.

```
scanf("%d", &value);
```

(e) (1 points) True or False: `strncpy()` adds additional null bytes to the destination, if the source string is less than the specified number of bytes copied.

(f) (2 points) What is the output displayed by the `printf()` calls on a 64-bit system?

```
int main() {
    int arr[] = {3, 2, 4, 1, 5};
    int *p = &arr[3];

    printf("%d ", *(p - ((p + *p)[-1])));
    printf("%lu\n", sizeof(arr));
}
```

(g) (5 points) Write a function named `flip_range()` that takes two arguments—an unsigned long, a start integer, and an end integer—and returns the result of flipping (toggling) the bits of the number in the first argument, in the range specified by the second and third integer arguments (uses zero-based indexing), both inclusive and valid. The function must use bitwise operations to achieve this. No credit will be given for other approaches.

(h) (2 points) What is the output displayed by `printf()` on a 64-bit, little endian system?

```
int main() {
    short arr[] = {0x1122, 0x3344, 0x5566, 0x7788};
    int *i_ptr = (int *)arr;

    printf("0x%x\n", *(i_ptr + 1));
}
```

(i) (2 points) Provide the `fopen()` statement for reading and writing a new file, called `new.txt`, that does not currently exist, and store its return value in a new variable named `fp`.

(j) (2 points) What is the value displayed by `printf()`?

```
char str[] = {'F', 'l', 'y', 0, 'H', 'i', 'g', 'h', 0};
printf("%s," "%lu\n", str + 5, strlen(str));
```

(k) (1 points) True or False: Function prototypes require the name of the function, its return type, the parameter names, and their respective types.

(l) (2 points) In the code segment, there are 5 different asterisk (*) characters. Rewrite the function, replacing all asterisks for square brackets ([]) wherever possible. If such a change is not possible (e.g. due to compilation error), leave the asterisk as is.

```
int *modifier(int *arr) {
    int *p = arr;
    return p + *(p + 2 - *p);
}
```

2. (15 points) Answer the following related question. Assume a 64-bit, little endian system.

   (a) (3 points) Declare a structure named `item_struct` which contains (in this order) an array of `MAX_BUF_LEN (21)` characters named `item_name`, and a short named `item_count`.

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

   (c) (2 points) Given the following code segment, what does `printf()` display?

```
item_t item_arr[10] = {0};
item_t *i2 = &item_arr[2];

printf("%lu %lu\n", sizeof(item_arr), sizeof(i2));
```

(d) (5 points) Declare a structure named `user_struct` which contains an array of `MAX_BUF_LEN (21)` characters named `username`, an array of `MAX_INVENTORY_SIZE (9)` `item_t` structures named `inventory`, an array of `TIME_FIELDS (2)` integers named `lastlog`, two floats named `health` and `hunger`, and a long named `user_id`. Order the elements of the structure such that the size of the structure is as minimized as possible. Simultaneously create a type `user_t` that refers to it.

(e) (3 points) Given the code segment, what does `printf()` display?

```
printf("%lu\n", sizeof(user_t));
```

3. (40 points) During the development of a game, you are tasked with taking in a save state file, and generating a human-readable report from it. Write a function named `state_report()`, that takes in an input file name (`char *`), and an output file name (`char *`), and should return `SUCCESS (0)` upon successful writing, and `ERROR (-1)` upon failure.

   These definitions, as well as `MAX_BUF_LEN (21)`, `MAX_INVENTORY_SIZE (9)`, and `TIME_FIELDS (2)`, are declared in a header file named `coords.h`. You must state all includes used, which includes the one used for the header file.

   The input file consists of a list of user data, with one user per line. Note that not every field will be used in the state report. The format of each line is given below:

   ```
   username;user_id|hour:min|date|savefile|health;hunger|inventory_count item(s)\inv1>n1|inv2>n2|...
   ```

   - `username[MAX_BUF_LEN]`: Name of the user.
   - `user_id`: Number that represents the user ID.
   - `hour, min`: Numbers that represent the last login hour ([0, 23]) and minute ([0, 60]), respectively.
   - `date`: Number in the format `YYYYMMDD`.
   - `savef`: Variable length string of alphanumeric characters that identify the user's data.
   - `health, hunger`: Floating-point numbers that represent the user's health and hunger statistics, respectively, in range of [0.0, 100.0].
   - `inventory_count`: Number that represents the number of items the user has in their inventory, assume that this number is always correct ([0, `MAX_INVENTORY_SIZE`]).
   - `inv`*`X`*`[MAX_BUF_LEN]`: Name of the *X* numbered item in the inventory (`inv1, inv2, ...`).
   - `n`*`X`*: Quantity that the user has of the *X* numbered inventory item (`n1, n2, ...`).

   Note that the number of items in the inventory equals the value in `inventory_count`. As such, you must handle a variable length of input, depending on the number of items that a user has, which is in the range of [0, `MAX_INVENTORY_SIZE`]. Beware that the word next to `inventory_count` may be different per input (`item` or `items`).

   A sample input file follows:

   ```
   Steve;1001|14:30|20240115|chjzl|98.5;76.3|3 items\strength potion>2|sword>1|shield>1
   Alex;1012|09:45|20231230|lc ivr|85.2;40.1|5 items\apple>13|bow>1|arrow>64|torch>43|map>1
   Eva;1042|23:59|20240101|p1vb52 himi|100.0;100.0|0 items\
   Diana;1104|07:15|20230720|14he8|65.4;20.7|2 items\iron pickaxe>1|diamond>5
   Jeff;2520|16:00|20240312|qs8m5|75.9;55.4|3 items\book>1|redstone dust>13|command block>3
   Chris;2400|22:10|20231005| 803tk|90.0;33.8|1 item\iron chestplate>1
   ```

   The output file should contain all the users separated by a new line, with their data related to their name, user ID, last login, health, hunger, total number of items in the inventory, and each individual inventory item name. Also, you must index every player in order of appearance in the input file, starting from index 1. Use one decimal place for floating-point numbers.

   ```
   User <index>: <usename>:<userid>
   Last login: <hour>:<min>
   Status: <health>:<hunger>
   Number of items: <number of items in inventory>
   Inventory:
   <item 1 name> -> <quantity of item 1>
   <item 2 name> -> <quantity of item 2>
   <item ... name> -> <quantity of item ...>
   ```

Part of a sample output file based on the sample input follows:

```
User 1: Steve:1001
Last login: 14:30
Status: 98.5:76.3
Number of items: 4
Inventory:
strength potion -> 2
sword -> 1
shield -> 1

User 2: Alex:1012
Last login: 09:45
Status: 85.2:40.1
Number of items: 122
Inventory:
apple -> 13
bow -> 1
arrow -> 64
torch -> 43
map -> 1

...
```

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 `ERROR`. Make sure to properly close any open files and set the file pointers back to `NULL`.

You may also use the structs and types created in question 2. Do not forget to assert the arguments of the function.

Do not write your answer on this page. Response will not be graded.

Work for question 3.

Continuation for question 3.

Continuation for question 3.

Continuation for question 3.

4. (20 points) You will need to work with a chunk structure that represents a chunk in a 3D grid system. Each chunk contains exactly eight structure vertex points defining its corners.

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

struct chunk {
    struct vertex corners[8];
};
```

Write a function named `transform_chunks` that applies a transformation to a group of chunks stored in a binary file. The function takes a file pointer that has been opened in read/write binary mode, and returns an integer of the number of vertices that have been modified.

The binary file begins with an integer, `num_chunks`, that indicates the number of chunks being transformed, followed by a `num_chunks` number of `struct chunk` structures. You should read each chunk, and call a function, `coord_mod()`, which takes in a `struct vertex` pointer, and returns `void`. Afterward, write the newly transformed vertices back into the file, replacing the original chunk.

You will need to write the function prototype for `coord_mod()` in its appropriate location, considering it was defined in a lower position in the C file.

Do not open nor close the file, and do not attempt to read all the chunks at once, as the file is significantly large. Assume no errors can occur.

Continuation for question 4.