CSE 1320: Intermediate Programming
University of Texas at Arlington
Spring 2021
Dr. Alex Dillhoff

Assignment 5

This assignment focuses on dynamic memory allocation.

Your code must compile without any warnings or errors and run without segmentation faults to receive credit. Points will be taken for inconsistent formatting.

1. Create a contains 10 functions which accomplish the subtasks described below. You must use dynamic memory allocation. No static arrays will be accepted.
   DO NOT FORGET TO FREE ANY ALLOCATED MEMORY!

   (a) Allocate space for a single integer and set the value to 20.
   (b) Allocate space for an integer array of size 200 and initialize the values to 0.
   (c) Allocate space for a 2D integer array of size (30, 40) and initialize the values to 0.
   (d) Allocate space for a 3D integer array of size (40, 40, 20) and initialize the values to 0.
   (e) Allocate space for a `double` and set the value to 3.14.
   (f) Allocate space for a `double` array of size 300 and initialize the values to 0.
   (g) Allocate space for a 2D `double` array of size (80, 80) and initialize the values to 0.
   (h) Allocate space for a `char` and set the value to `0`.
   (i) Allocate space for an `char` array of size 2048 and initialize the values to 0.
   (j) Allocate space for a 2D `char` array of size (1024, 1024) and initialize the values to 0.

   Save your code as `problem1.c`.

2. Create a program which takes two integers as input from the command line. In your code, allocate space for a 2D array dependent on user input. For example, if they enter `30 40` then your program should allocate space using `double **`.

   Once the memory is allocated, initialize the data by setting the value at index (i, j) equal to its position as a 1D index ($i * M + j$, where M is the number of columns).

   Save your code as `problem2.c`.

3. Create a program that reads in Chirp data from a raw CSV file. Your program should determine how many entries are in the file and allocate the appropriate amount of memory. Each entry should be stored in a `struct` which matches the member data given. After the data is read and parsed, print the file contents in

a formatted table as seen in the example run. Note that message data must be allocated dynamically. A sample file can be found on Canvas.

Requirements

- Your Chirp `struct` should be declared in a header file with a header guard. Any functions created must have a declaration in the same file.
- Read in file name from command line.
- Check that the file exists.
- List all contents as seen in the example run.
- Free any memory allocated.

Chirp Data

- User ID `int`
- Date `char array of size 8`
- Message `char *`

Example Run

```
$ ./read_chirps chirp_data.csv
On 2021/01/27 @amos Chirped,
"It's better to go down swinging than rolling over"
...
```

Your code should be saved as `problem3.h` and `problem3.c`.

4. Machines have far more file storage space than memory space. Because of this, applications will only load in resources that are necessary given the context. A game developer has written creature data to a binary file given the data format below. Your task is to create a program which only loads the creatures of a requested zone. Your program should allocate memory for the data ONLY when it is requested. Once loaded, display the data as shown below.

   Requirements

   - The creature `struct` should be declared in a header file with a header guard. Any functions created must have a declaration in the same header file.
   - Accept two command line arguments for the file name and requested zone.
   - Search through the file and allocate memory for each creature that is in the requested zone. The creatures should be loaded in an array that is dynamically allocated as well.
   - Once the end of the file is reached, print out the creatures found.
   - Use the file `creatures.db` on Canvas to test your program. There are only 3 zones (1, 2, 3) in the file.

   Data Format

   - `long` size

- `int` zone
- `int` cr
- `int` ac
- `int` hp
- `char` * name

Example Output

```
$ ./read_creature c.db 3
Name: Adult Bronze Dragon
CR: 15
HP: 212
AC: 19
Zone: 3
Name: Adult Red Dragon
CR: 17
HP: 256
AC: 19
Zone: 3
```

Create a `zip` file using the name template `LASTNAME_ID_A5.zip` which includes the all required code files. Submit the `zip` file through Canvas.