CS 277 Lab 5
Pointers and Debugging
Due 11:59pm Sunday March 30, 2014

Purpose

The purpose of this lab is to dig a little deeper with pointers through using the GDB debugging tool.

Assignment

Help! I had to write a Collection library and accompanying main() routine to demo the functionality. Unfortunately, I was up all night with a sick baby and my brain ended up fried. As a result my precious code is riddled with errors and I don't have time to track them down! Your job is to fix the mistakes in my **`collection.c`** and **`driver.c`** files so that the demo runs flawlessly. Use GDB to track down the problems. The **`output.txt`** file demonstrates what the output of the demo program should look like once you have tracked down all of the bugs. Oh. And another thing. I think there might be a memory leak. Or Two. Whoops. You will also need to figure out where any memory leaks might be and fix those too. Use the provided makefile to compile the program. The demo code is compiled into an executable named **`demo`**. Once you have it running, you might also want to use GDB to step through the code and double check that everything is correct even if the output matches what is in the sample file. Just to be sure.. cause you never know. Please include a file named **`changes.txt`** where you enumerate the changes/fixes you made and what error the change addressed. To start GDB with the executable run:

```
gdb ./demo
```

To give you an idea of what the collection library does, it defines a data type called Collection and the set of operations that can be performed on the Collection. Check out **collection.h** to see the data type definitions and the function prototypes. The Collection object maintains a doubly-linked list (Each element in the collection maintains a pointer to both the previous element and the next element in the Collection). New elements are added to the end of the list. The operations on Collections are as follows:

```
Collection new_collection();                    /// allocates memory for and initializes a collection
void destroy_collection();                      // frees memory used by the collection object
int collection_add_item(Collection, void *);    // adds a new item to the Collection, returns the new
                                                // size ( # of items) of the Collection.
void * collection_item_at(Collection, int);     // returns a pointer to the item in the Collection at the
                                                // 0-indexed location
void collection_remove_at(Collection, int);     // removes from the Collectio the item at the 0-indexed
                                                // location
void ** collection_as_array(Collection);        // returns the objects stored in the Collection in an
                                                // array
```

Submission

To submit, generate a .tar file named lab5-yourusername.tar containing your solutions for driver.c and collection.c as well as your changes.txt file. Email the .tar file to mharmon@lclark.edu before the due date and time.

Hints and Tips

- GDB will not (directly) tell you about memory leaks, however, it does provide you with the tools you need to make well-educated guesses as to where they might be.
- Use GDB to your advantage. This lab can be solved with relatively minor code changes if you take the time to experiment with GDB and understand the different tools it provides.

Evaluation

+15: compiles **+20:** runs **+50:** runs & changes to library+driver produce correct output/no errors
+10: "hidden" problems not obvious via running the demo are corrected.
+5: memory leaks have been abolished.