

# Homework4 (code)

- In this homework, you will gain some familiarity with memory allocation.
  - First, you'll **write some buggy programs (fun!)**.
  - Then, you'll use some tools to help you find the bugs you inserted.
  - Then, you will realize how awesome these tools are and use them in the future, thus making yourself more happy and productive.
  - The tools are the debugger (e.g., gdb), and a memory-bug detector called valgrind.

# Questions

- 1. First, write a simple program called `null.c` that creates a pointer to an integer, sets it to `NULL`, and then tries to dereference it. Compile this into an executable called `null`. What happens when you run this program?
- 2. Next, compile this program with symbol information included (with the `-g` flag). Doing so let's put more information into the executable, enabling the debugger to access more useful information about variable names and the like. Run the program under the debugger by typing `gdb null` and then, once `gdb` is running, typing `run`. What does `gdb` show you?

# Questions

- 3. Finally, use the valgrind tool on this program. We'll use the memcheck tool that is a part of valgrind to analyze what happens. Run this by typing in the following: `valgrind --leak-check=yes null`. What happens when you run this? Can you interpret the output from the tool?
- 4. Write a simple program that allocates memory using `malloc()` but forgets to free it before exiting. What happens when this program runs? Can you use `gdb` to find any problems with it? How about `valgrind` (again with the `--leak-check=yes` flag)?

# Questions

- 5. Write a program that creates an array of integers called data of size 100 using malloc; then, set data[100] to zero. What happens when you run this program? What happens when you run this program using valgrind? Is the program correct?
- 6. Create a program that allocates an array of integers (as above), frees them, and then tries to print the value of one of the elements of the array. Does the program run? What happens when you use valgrind on it?

# Questions

- 7. Now pass a funny value to free (e.g., a pointer in the middle of the array you allocated above). What happens? Do you need tools to find this type of problem?
- 8. Try out some of the other interfaces to memory allocation. For example, create a simple vector-like data structure and related routines that use `realloc()` to manage the vector. Use an array to store the vectors elements; when a user adds an entry to the vector, use `realloc()` to allocate more space for it. How well does such a vector perform? How does it compare to a linked list? Use `valgrind` to help you find bugs.