| CSE 3100 Systems Programming | Spring 2024 |
|---|---|
| Lab #3 | |

Welcome to Lab 3! Read each question carefully before beginning work; Submit your code to gradescope.

We are going to learn another useful tool `valgrind`, which helps you to find bugs in programs. The official site of `valgrind` has a quick start guide.

**Part 1. squares.** Use `gdb` and `valgrind` to find and correct all compiler warnings and bugs in `squares.c`. You may need to look at the `man` pages for any library functions used in the code (e.g., `atoi()`) to understand their function and to ensure they are being used correctly. This exercise provides us an opportunity to debug other people's code. Completely rewriting the program will not let us learn as much.

**Part 2. argvcat.** `argvcat` is a program that concatenates all the arguments in `argv` to a single string and prints out the concatenated string. Starter code is provided in `argvcat.c`.

The goals of this assignment is to 1) complete the function `my_strcat()` and 2) fix all memory leaks in the code (Of course, you just learned `valgrind`.)

The function `my_strcat()` has the following prototype:

The function takes two strings, `s1` and `s2`, as parameters and returns a new string that is the concatenation of `s1` and `s2`. The memory space for storing the return string should be dynamically allocated from the heap. The function should request the least amount of space that is needed to hold the result. The function should not change the strings `s1` or `s2`. For example, the function should not try to free the memory used by `s1` or `s2` because 1) their memory space may not be dynamically allocated, and 2) the caller may want to keep `s1` or `s2`.

You can use string functions in C library, for example, `strlen()` and `strcat()`. Read `man` pages to learn them.

If `malloc()` fails, call `my_error()` to print an error message and exit.

It is necessary to add code in the `main()` function to free memory. However, the changes should not affect how `my_strcat()` is called and how the loop is constructed.

The program should work as in the sample runs listed below after `my_strcat()` is implemented, even before memory leaks are fixed. The final program should not have memory leaks.

```
$./argvcat
./argvcat
$./argvcat a 1 b 2 234
./argvcata1b2234
$./argvcat 1 a '' '#.' and more arguments
./argvcat1a#.andmorearguments
```