CS270 Recitation 6
"C Pointer Exercise"

Goals

To help you learn and improve your understanding of C pointers, including single and double pointers, static and dynamic memory, basic data type arrays (ex: arrays of ints), and complex data type arrays (i.e., arrays of structs).

The Assignment

Make a subdirectory called R6 for the recitation assignment and copy the following files into the directory:
http://www.cs.colostate.edu/~cs270/Recitations/R6/pointers.c
http://www.cs.colostate.edu/~cs270/Recitations/R6/pointers.h
http://www.cs.colostate.edu/~cs270/Recitations/R6/Makefile

Find a partner and form into groups of two, and open up `pointers.c` and `pointers.h` in `gedit`. The TA will explain a few basics about C pointers. Next, you will need to add code to `pointers.c` according to the instructions found in the comments inside `pointers.c`.

You will be given a short amount of time to complete each task. At the end of this time, the TA will give out the answer, or ask one of you to solve the problem on the white board. However, you are expected to work on the problems during the allotted time. If you fail to make an effort, you will loose the grade for attending the recitation.

Description of the functions you will be working on:

```
/*
 * a function whose input is passed by value
 * - demonstrates how passing by value does not change the original value
 */
void pass_by_value(int var);

/*
 * a function whose input is passed by reference
 * - demonstrates how results can be returned using pass by reference
 */
void pass_by_reference(int *ptr);

/*
 * print pointer variables and their values
 * - demonstrates the use of single and double pointers
 */
void print_pointers();

/*
 * print the bits of a float variable using pointer manipulation
 * - demonstrates typecasting and pointer magic
 */
void print_float_bits(float f);
```

```c
/*
 * go through an array of ints using array indices and pointer arithmetic
 * - demonstrates how C arrays are really pointers undeneath.
 * - understand how to staticly and dynamically allocate memory
 */
void arrays_and_pointers(int size);

/*
 * deliberately cause segfaults and memory corruption
 * - understand why and how these are caused
 */
void memory_corruption();

/*
 * create both static and dynamic array of structs (complex data types)
 * - understand the syntax involved in working with arrays of structs
 */
void array_of_structs(int size);
```