CS270 Recitation 7
"C Pointer Exercise"

* This is a repeat of R6. We will continue where we left off (i.e., from function 4 onwards)

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 R7 for the recitation assignment and copy the following files into the directory:
http://www.cs.colostate.edu/~cs270/Recitations/R7/pointers.c
http://www.cs.colostate.edu/~cs270/Recitations/R7/pointers.h
http://www.cs.colostate.edu/~cs270/Recitations/R7/Makefile

Open up `pointers.c` and `pointers.h` in `gedit` and implement the three functions described below. Note that we covered these in the last recitation (i.e., R6), so you should at least make an attempt at completing them (if you are unable to do so, the TA may ask you to form a separate group, and you will loose the point for attending the previous recitation).

```c
/*
 * 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();
```

After you finish implementing the above three functions, the TA will implement the following functions and example the operations in each step. However, you are urged to practice these problems on your own, at a later time, and verify your solution against the one provided by the TA.

```c
/*
 * 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);
```