Exercise #7
Due Friday, 02/19/2016, at 11:59pm

(Note: If you don’t know the answer to a code-related question, try writing a small program to test things out!)

Vocabulary
Give a brief definition of each of the following terms:
- Heap
- Dynamic Array
- The ‘new’ Keyword

Pointers and Arrays
After the following code has been executed:

```c
int arr[5] = {3, 2, 7, 10, 1};
int *ptr = arr;
```

What will the value of each of the following expressions be?

- a) `arr[2]`
- b) `ptr[2]`
- c) `*arr`
- d) `*ptr`
- e) `*(ptr+3)`
- f) `*(arr+3)`
- g) `*ptr+1`
- h) `*arr+1`

What conclusions can you draw about the relationship between pointers and arrays from these results?

Arrays and Functions
Consider the following function which prints out the fourth element in an array:

```c
void print_fourth(int a[4]) {
}
```

Will the following code compile? If so, what will it print out?

```c
int array[5] = {1, 2, 3, 4, 5};
int too_small[2] = {10, 20};
print_fourth(array);
print_fourth(too_small);
```

Now, assume we changed the ‘a’ parameter in `print_fourth` into a pointer to an integer instead of an array of integers. How would that change your answer to the previous question?

Incorrect Array Usage
The function below is supposed to create an array, initialize it, and return it to the calling function.

```c
int* make_array(int size) {
    int array[size];
    for (int i = 0; i < size; i++) array[i] = i;
    return array;
}
```

Unfortunately, the code above is broken. What’s wrong with it, and what needs to be done to make the function work as intended?

Working With C Strings
You’ve mostly worked with C++ strings so far (i.e. variables with a `string` type), but it’s important to know how C-style strings function as well. C-style strings are just an array of characters which must always be end with a null character.

For example one way to properly define a C string is as follows:

```c
char cstr[4] = {'H', 'i', '!', '\0'};
```

Note that the array required to hold a string with three printable characters must be four elements long, since a null character (‘\0’) must be at the end!

Based on everything you’ve seen above, write a function that will calculate the number of printable characters in a C-string (i.e. the length of the string) that matches the following function signature:

```c
int my_strlen(char *c);
```

(Note: The function’s body shouldn’t be longer than about five lines!)

For take-home exercises completed in peer-led groups, each student must participate in the class discussion and write answers to each of the questions on his/her own paper to show for credit.

For take-home exercises completed on your own, turn in your work electronically using the TEACH website.