ENCM 339 Fall 2016 Midterm: Solutions for parts c and d of Section 3

Part c

Using character constants such as `'a'` and `'z'` is easier than doing arithmetic with ASCII codes!

```
int count_lower_cases(const char *str)
{
  int count = 0, i;
  for (i = 0; str[i] != '\0'; i++) {
    if (str[i] >= 'a' && str[i] <= 'z')
      count++;
  }
  return count;
}
```

Part d

For most strings, the goal is to put a `'\0'` character just after the last non-space character in the string. The string gets shortened by putting in a new `'\0'` character, not by moving characters around in the array where the string lives.

There are a couple of special cases to be handled: an empty string, and a string that is composed of nothing but spaces.

This solution finds the end of the string, then walks back to find the last non-space character:

```
void trim_trailing_spaces(char *str)
{
  int len;
  len = strlen(str);
  if (len == 0)
    return;
  int i = len - 1;

  // Check i >= 0, in case string is nothing but spaces.
  while (i >= 0 && str[i] == ' ')
    i--;
  str[i + 1] = '\0';
}
```

This solution just walks the string from start to end, updating an index whenever a non-space character is found:

```
void trim_trailing_spaces(char *str)
{
  // This initialization handles an empty string
  // and also a string that is nothing but spaces.
  int last_non_space = -1;

  for (int i = 0; str[i] != '\0'; i++)
    if (str[i] != ' ')
      last_non_space = i;

  str[last_non_space + 1] = '\0';
}
```