Lab exercise 6

1. **Write a program to concatenate two character strings without passing the number of characters in each string as arguments. You can modify the function in Lecture 11 as:**

   ```
   void concat (char result[], const char str1[], const char str2[]);
   ```

2. **In C, you cannot directly test two strings to see if they are equal with a statement such as if (string1 == string2) ... because the equality operator can only be applied to simple variable types. Write a program to determine if two strings are equal, the head of function is defined as below.**

   ```
   bool equalStrings (const char s1[], const char s2[])
   ```

3. **Consider a practical text processing application: counting the number of words in a portion of text. The program has a function called countWords(), which takes as its argument a character string and which returns the number of words contained in that string.**

   ```
   int countWords (const char string[])
   ```

   The main function is defined as follows:

   ```
   int main (void)
   {
   const char text1[] = "Well, here goes.";
   const char text2[] = "And here we go... again.";
   int countWords (const char string[]);
   printf ("%s - words = %i\n", text1, countWords (text1));
   printf ("%s - words = %i\n", text2, countWords (text2));
   return 0;
   }
   ```

4. **Write a function called substring() to extract a portion of a character string. The function should be called as follows:**

   ```
   substring (source, start, count, result);
   ```

where source is the character string from which you are extracting the substring, start is an index number into source indicating the first character of the substring, count is the number of characters to be extracted from the source string, and result is an array of characters that is to contain the extracted substring. For example, the call:

substring ("character", 4, 3, result);

extracts the substring "act" (three characters starting with character number 4 from the string "character" and places the result in result.

Be certain the function inserts a null character at the end of the substring in the result array. Also, have the function check that the requested number of characters does, in fact, exist in the string. If this is not the case, have the function end the substring when it reaches the end of the source string. So, for example, a call such as substring

("two words", 4, 20, result);

should just place the string "words" inside the result array, even though 20 characters were requested by the call.