Lab Exercise - Week 12

This week we are going to work with strings. First, without using the string library functions and then using the string library functions. See this very short introduction to the string functions available in the C library:https://www.tutorialspoint.com/cprogramming/c_strings.htm

Experiment 1a.
Program Name: substring1.c

Write a C program (without using the string library functions) to read two strings and check whether the first string occurs as a substring of the second.

Experiment 1b.
Program Name: substring2.c

Use the string library function strstr(). See the example below.

```
/*
Program to check if one string is contained as substring in another
*/


#include <stdio.h>
#include <string.h> /* To include the string functions like strstr() */


int main()
{
  char small[100], big[200];
  char *p;
  int offset;


  printf("Enter the big string: ");
  scanf(" %[^\n]s", big);
  /* the " %[^\n]s" is used to scan until you see a new line (enter key).
  So you can scan strings with more than one word using one scan statement %/


  printf("Enter the small string: ");
  scanf(" %[^\n]s", small);


  /* strstr(big, small) returns 0 if the small string is not contained in the big str
ing.
```

```
  Otherwise, it returns a character pointer to the starting location of the first
  occurence of the small string in the big string */
  p = strstr(big,small);


  if(p == 0)
  printf("%s is not contained in %s\n", small, big);
  else
  {
    offset = (int) (p - big);
    printf("%s is contained in %s, starting from position %d\n", small, big, offset);
  }


  return(0);
}
```

Modify the above program to make the search **case-insensitive**. That is the small string "rat" is contained in the big string "Rat is a rodent." It should work even if the small string was "RAT", "Rat", "raT" etc.

Hint. You may convert both strings to lowercase using the tolower() function available in the library ctype (#include <ctype.h>). But remember that tolower can convert only one character at a time.

Experiment 2a:
Program name: search_file1.c

Write a program to scan a word (no spaces, commas, etc) and a filename and then check whether the word is contained in the file as a substring.

You can use this "quotes.txt" file for testing (Download it your working directory). Search for "psychopath" - it should succeed and search for human and it should fail.

Note that Linux has a built-in program that does the above job. It is called "grep". You can run it by

```
grep psychopath quotes.txt
```

The program grep is actually much more powerful than this. In fact, that is why it is called "GNU Regular Expression Search" and not just "search" :)

Experiment 2b:
Program name: search_file2.c

Modify the above program so that **only whole words** are matched. For example, if the input word is "program" then it should NOT match the words "programming" or "programmer" or "programs" in the file. Also take care of punctuation marks. If the file contains "program." or "program," (with a full stop or comma), then it should match.

The C library function **strcmp(str1, str2)** compares the string pointed to by str1 to the string pointed to by str2. The function always returns an integer. if the two strings are the same, then the return value is 0.

Test your program with the words "program" and "civilization"

Modify the so that the search is **case-insensitve**.

Homework

Experment 1:
Program name: spell_checker.c

Build on the previous experiment to make a spell-checker. You can use the following "spell.words" file which contains the most commonly used 50,000 English words (with their correct American spelling). You can assume that a word in the input file (for example quotes.txt) is wrongly spelt if (the lower case version of) that word is not found in the file "spell.words". Your program should output all the wrongly spelt words in the input file (one word per line).

Homework 2:
Program name : find_replace.c

Write a program to scan two words (first and second) and two filenames (input and output) and then replace every occurence of the first word in the input file (as a complete word) with the second word. The output should be written to the output file.

Homework 3:
Program name: search_file3.c

Modify search_file1.c so that you can string for strings which contains multiple words. That is you can search for strings like "best thing" and "only the third" in the input file (quotes.txt).

You may need to learn the fseek function so that you move back your file pointer when needed. See https://www.tutorialspoint.com/c_standard_library/c_function_fseek.htm

Happy Coding

---------- *All the best* ----------