School of Computer Science, University of Windsor

60-141: Introduction to Algorithms and Programming II
Term: Summer 2014 (July-August)
Instructor: Dr. Asish Mukhopadhyay

Lab 4
Posted: 24th July, 2014
Due: Beginning of lab5

Preamble: This lab is designed to consolidate your understanding of structures, array of structures, array of pointers to structures and string manipulation. This will require a thorough understanding of the programs discussed in the lectures. Each one of the programs should be properly commented, following the style of your textbook. All lab work is expected to be original.

Grading Scheme: Each problem is worth 10 points, for a total of 20 points. The scoring break-up for each problem is: 2 points for program documentation, 2 points for effort and 6 for correctness.

Credits: The conception of this lab is original and due to me.

Problem 1(a) Consider the following function definition that checks if t is a `substring` of s:

```
/* strindex: return index of t in s, -1 if none */
int strindex(char s[], char t[])
{
   int i, j, k;

   for(i = 0; s[i] != '\0'; i++) {
      for (j = i,  k = 0; t[k] != '\0' && s[j] == t[k]; j++, k++)
            ;
      if (k > 0 && t[k] == '\0')
         return i;
   }
   return -1;
}
```

Write a modified version of the above function, `strrindex(s,t)`, which returns the position of the rightmost occurrence of t in s , or -1 if there is none. Test your function by calling it from a main

function.

(b) For the same assumptions as in part (a), write a function `strend(s, t)` which returns 1 if t occurs at the end of s and 0 otherwise. Test your function by calling it from a main function.

(5 + 5) points

Problem 2: Given a text file as input, print out the distinct words that appear in the file, along with a count of their frequencies. Thus it is a generalization of an earlier lab problem of printing out the distinct integers of a set of input integers, assuming that space is at a premium. The difference is that we are dealing with words instead of integers. Thus if the text is:

```
I am working hard to solve the lab problems;
the problems are hard to solve
```

the frequency table is:

word	frequency
I	1
am	1
working	1
hard	2
the	2
to	2
problems	2
solve	2
are	1
lab	1

Use the program framework that we discussed in class to solve the problem of counting the frequency of the keywords in a C-program. This means that you have to define a structure with two fields - one a pointer to a word and the other a count of its frequency of occurrence as below.

```
struct key {
    char *word;
    int count;
} wordtab[MAXSIZE];
```

Solve the problem both by maintaining an array of such structures, as well by an array of pointers to these structures, just as we did for the keyword-counting program. Retain the definition of a word used in the keyword counting program so that you can use the getword() function as it is. You can set the maximum length of a word (MAXLEN) and the maximum number of words (MAXSIZE) by using the define directives.

You will also need the following function `dupStr(word)`

```
char *dupStr(char *s)
{
        char *p;
        p = (char *) malloc(strlen(s) + 1); /* +1 for \0 */
         if (p!= NULL)
             strcpy(p,s);
         return p;
}
```

to copy a new word into the appropriate field of a `wordtab` entry.

(10 points)