DALHOUSIE UNIVERSITY
Inspiring Minds

Fall 2017 (Sep5-Dec5)
Faculty of Computer Science

# CSCI 2132 — Software Development

## Assignment 7

---

**Due**: *Monday, Nov 27, 2017 by 11:59 p.m.*
**Instructor**: Vlado Keselj, CS bldg 432, 494-2893, vlado@dnlp.ca

---

**Assignment Instructions**:

Solutions to this assignment must be submitted through SVN, in a similar way as for the previous assignment.

The first question refers to the Practicum 1 and your results from the practicum will be counted as marks for this questino.

The answers to all other questions must be submitted in your SVN directory: *CSID*`/a7` where *CSID* is your CS userid. Remember that you need to add to `svn` the directory as well as any files that you want submitted.

**1)** (30 marks) The marks for this question will be assigned according to the results of Practicum 1.

**2)** (12 marks) Write answers to this question in the plain text file named `a7q2.txt` and submit using SVN in the directory: `a7`

You are given the following code:

```
char a[] = "This is a sentence.\n";
char *p = a;

strcpy(p+5, "ABCD");  /* line 1 */

strcat(a, "0123");    /* line 2 */

*(p+strlen(a)) = '-'; /* line 3 */
```

Show the following informatino after each of the lines labelled as 'line 1', 'line 2', and 'line 3':
– full contents of the array `a`

– the content of the array `a` if it is treated as a string and
– the length of the string that would be returned by the function `strlen(a)`.

When showing the content of the array, use the following codes for invisible characters: `\x20` for space, `\x0A` for newline, and `\x00` for the null character and separate all characters by space, and be sure to show all 21 elements of the array.

When showing `a` as a string, delimit the contents with double-quote characters (`"`) and show the content only until the null character, as normal for the strings. Do not use the above codes for strings (see the example below).

For example, the required information shown just before 'line 1' is:

```
a: T h i s \x20 i s \x20 a \x20 s e n t e n c e . \x0A \x00
a as string: "This is a sentence.
"
length of a: 20
```

**3)** (20 marks) Write a C program named `a7q3.c` which reads the standard input and produces a list of all distinct words in a lexicographic order. The program must be submitted in SVN in a separate directory for this question with the following relative path in your SVN repository: *CSID*`/a7/a7q3/` This means that the relative path of the C program file will be: *CSID*`/a7/a7q3/a7q3.c` where *CSID* is the root directory of your SVN repository.
**Be careful** to submit the program in the right directory, otherwise you will receive 0 marks for the question.

You also must submit the files `test1.in`, `test1.out` `test2.in` and `test2.out` that can be used for testing your program. The files `test1.in` and `test1.out` are files provided in the directory `~prof2132/public` under the names `a7q3-test.in` and `a7q3-test.out` and you will need to rename them. The files `test2.in` and `test2.out` must be prepared by you. These files contain standard input and expected standard output of your program.

For example, if you type:

```
gcc a7q3.c
./a.out < test1.in > test1.new
diff test1.out test1.new
```

there should be no differences reported, since your program output saved in `test1.new` should be the same as `test1.out`.

Now, let us go back to the program. The program will read the standard input and produce a list of all distinct words to the standard output in a lexicographic order. We will consider a word to be any sequence of letters. All words must be transformed to the lowercase letters. You can assume that there are no more than 10,000 distinct words, and that each word is not longer than 25 characters.

The program should first read all the input. If it finds any words longer than 25 letters, such words should be ignored; and if it finds more than 10,000 distinct words after making the words lowercase, all additional words should be ignored.

After reading input, the list of words must be printed to the output in the lexicographic order, one word per line.

If any words were ignored because they were longer than 25 characters, the following warning message should be printed at the end:
`WARNING: Words longer than 25 characters were ignored.`

Similarly, if there were more than 10,000 distinct words, the following warning message should be printed at the end:
`WARNING: More than 10,000 words.  Some words are ignored.`

If any of the above two warnings are printed, they must be separated by a blank line from the previous list of words, and if both warnings are printed, the first one (about 25 characters) should be printed first in a line, immediatelly followed by the second warning on the next line.

### Sample Input

```
This is a sample input.
The program should first read all the input.  If it found any words
longer than 25 letters, such words should be ignored; and if it found
more than 10,000 distinct words after making the words lowercase, all
additional words should be ignored.

A very long word: PNEUMONOULTRAMICROSCOPICSILICOVOLCANOCONIOSI
```

### Sample Output

```
a
additional
after
all
and
any
be
distinct
first
found
if
ignored
```

```
input
is
it
letters
long
longer
lowercase
making
more
program
read
sample
should
such
than
the
this
very
word
words

WARNING: Words longer than 25 characters were ignored.
```