IT483 Fall 2023
Lab 1 – Part 2

Total points: 30 programming points

The main objective of Lab 1- part2:
Edit, Write, compile, and run simple C programs.

Submit your programs (question#2-5) to ReggieNet website for IT483.
- **Submit a SINGLE "ZIP" FILE**
- The single ZIP file should contain **C source files and Makefiles** in the following subdirectories appropriately: 2, 3,4,5,6
- You might want to use the following Linux command to create a single zip file:
  % zip –r ***yourLastName_yourFirstName_lab1part2*.zip** ./2 ./3 ./4 ./5 ./6

Questions: (Regarding C program language)

1. (0 points)

   (a) Download PDF file for "Laboratory tutorial" from the website for OSTEP
      a. https://pages.cs.wisc.edu/~remzi/OSTEP/lab-tutorial.pdf
      b. Start to read the document.

   (b) Download PDF file for "An introduction to C programming for Java Programmers" written by Mark Handley at UCL (http://www.site.uottawa.ca/~mgarz042/CSI3131/files/c-introHandley.pdf). Save the downloaded PDF file in your H: drive. You will need to use this PDF as a reference book for C programming language.

   (c) Download lecture slides files from http://www.cs.cornell.edu/courses/cs2022/2011fa/ and store them in your ISU remote datastore drive (i.e., H drive or Office365 drive)
      a. You may use the lecture slides as a reference for C programming language.
      b. Virtual workshop for C from Cornell: **https://cvw.cac.cornell.edu/Cintro**

   (d) Download C language cheat sheet(s) from Week 2 module on ReggieNet website for IT483 and store it in your ISU remote datastore drive (i.e., H drive or Office365 drive)

   (e) You should have started to read K&R C language programming.

   Deliverables: NONE

2. **(5 points) [Example: File I/O]**

   Create a subdirectory called "2". Change the current directory to "2".
   Type in two separate C programs in page 31 and 32 (in "An introduction to C programming for Java Programmers").
   You will also need to do additional work to make the C programs work correctly.
   Examples are
   - *Add #include .. statement(s) (i.e., you need to include appropriate header file(s))*
   - *Add main() {…}*
   - *Change "/tmp/foo" to "./yourname_foo" to avoid conflicts*

Refer to the documents that you downloaded while you were working on Question#1.
Write a Makefile that automates the compilation of the two C programs. (Note that the Makefile should specify the dependency between binary files and C source code.)
Then, compile and execute the programs using appropriate data files.

Deliverables: C source file(s) and Makefile

3. (5 quiz points) [Example: malloc]

Create a subdirectory called "3". Change the current directory to "3".
Type in the C program in page 17 (An introduction to C programming for Java Programmers). Write a Makefile that automates the compilation of the C program. Then, compile and execute the program.

Deliverables: C source file(s) and Makefile

4. (5 points) [Arrays and files]

Create a subdirectory called "4". Change the current directory to "4".
Write a program to read ten integer numbers from the keyboard, and store them in an array of integers. Then, print out the numbers in the sorted order (in increasing order) on the screen and also store the sorted (in increasing order) numbers in a text file called "sorted.txt". Note: you need to use fopen/~~fread~~/fclose APIs. You may use (sprint AND fwrite) or fprintf for writing data into a file in the correct data format. Note that the numbers should be stored in a text file in PLAIN TEXT FORMAT. In other words, your program needs to convert each integer number to ASCII codes before it is stored in a text file.
You may implement any "sorting" algorithm (e.g, bubble sort, quick sort etc)

For example, if the user types in
10000
11
12
13
14
15
16
17
1800
19

Then,
The following should be printed on the console and stored in the output text file in ASCII format (so that we can open the file using notepad and visually see the numbers successfully)
11
12
13
14
15
16

17
19
1800
100000

Deliverables: C source file(s) and Makefile

5. **(5 points) [Arrays and files -revisited]**
Create a subdirectory called "5". Change the current directory to "5".

Write a program to read ten integer numbers from the keyboard, and store them in an array of integers. Then, print out the numbers in the sorted order (in decreasing order) on the screen and also store the sorted (in decreasing order) numbers in a text file called "sorted.txt". . Note: you need to use all of the following C APIs: open, write, sprint, and close. (DO NOT use fopen/fwrite/fclose APIs). Note that the numbers should be stored in a text file in PLAIN TEXT FORMAT. In other words, your program needs to convert each integer number to ASCII codes before it is stored in a text file.
You may implement any "sorting" algorithm (e.g, bubble sort, quick sort etc)
You want to check the following tutorial: http://www.techytalk.info/linux-system-programming-open-file-read-file-and-write-file/

Deliverables: C source file(s) and Makefile

For example, if the user types in
10000
11
12
13
14
15
16
17
1800
19

Then,
The following should be printed on the console and stored in the output text file in ASCII format (so that we can open the file using notepad and visually see the numbers successfully)
10000
1800
19
17
16
15

14
13
12
11

6. **(10 points) [Files and Strings]**
Create a subdirectory called "6". Change the current directory to "6".
Write a program that prompts the user for the name of a file. Then it opens the file, and counts the number of words and lines in the file, and print out those counts. Try to make use of functions to break the code into suitable functional blocks – for example, you might use a function to count the number of words in a line.

Your program must use strtok() API which is presented in the following:

(**Hint from http://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm)** strtok() is a little more complicated in operation. If the first argument is not NULL then the function finds the position of any of the second argument characters. However, the position is remembered and any subsequent calls to strtok() will start from this position if on these subsequent calls the first argument is NULL.

`char *strtok(char *s1, const char *s2)` -- break the string pointed to by `s1` into a sequence of tokens, each of which is delimited by one or more characters from the string pointed to by `s2`.

For example, If we wish to break up the string str1 at each space and print each token on a new line we could do:

```
---------------------------------------------------------------------------------------------
#include <stdio.h>
#include <string.h>
char str1[] = "Hello Big Boy     !!!";
char *t1;

main()
{

        char *str;
        str = (char *) malloc(strlen(str1)*sizeof(char)+1);

           for ( t1 = strtok(str,"  \t\n");  t1 != NULL;   t1 = strtok(NULL, " \t\n") ) {
                  printf("%s\n",t1);

           }
 }
```

First iteration: t1 = “Hello”

“Big Boy !!!”

T1 = “Big”

“Boy !!!!”

Strtok(str1,” \t\n”)

Hello xyz

World guys

Here we use the for loop in a non-standard counting fashion:

- The initialization calls `strtok()` loads the function with the string `str`
- We terminate when t1 is `NULL`
- We keep assigning tokens of `str` to `t1` until termination by calling `strtok()` with a `NULL` first argument
- Note that you can specify multiple delimiters for strtok(), e.g.,
  - Strtok(str,” \t\n”);
  - Space character(s), tab(s), and line feed characters are delimiters in this example.
- More details: https://www.geeksforgeeks.org/strtok-strtok_r-functions-c-examples/
- Also you may use strsep:
  - https://c-for-dummies.com/blog/?p=1769

For your information, you can use “wc” command to check if your program correctly counts the number of words in a given text file. (https://www.tecmint.com/wc-command-examples/)

Deliverables: C source file(s) and Makefile

***Acknowledgments**: The content of Linux tutorial is adapted from Unix Lab Tutorial written by Chung-Chi Li at IT. The content of C tutorial is adapted from C programming exercises written by Mark Handley at UCL.*

[HINT] How to create a makefile:

1. The name of the makefile should be "Makefile" (case-sensitive).
2. In general, "Makefile should be located in the directory where your "*.c" files are located.
3. If you have a single .c file (say, test.c), the content of the Makefile should look like this:
   [first line] **test: test.c**
   [second line] **[tab]gcc –o test test.c**

   If you have two separate programs (say, prog1.c and prog2.c), the content of the Makefile should look like this:
   [line 1] **all: prog1 prog2**
   [line 2]
   [line 3] **prog1: prog1.c**
   [line 4] **[tab]gcc –o prog1 prog1.c**
   [line 5]
   [line 6] **prog2: prog2.c**
   [line 7] **[tab]gcc –o prog2 prog2.c**
4. **You simply type "make" command to compile your code using "Makefile"**

[Hint] http://www.techytalk.info/linux-system-programming-open-file-read-file-and-write-file/