Data Structures & Algorithms
Sem II 2013-14
CS/IS F211
Lab 1
Topics: C Programming - Command Line Arguments, File I/O, Linked Lists.

_____

1. Command Line Arguments [Expected Time: 30 minutes]

A simple program in C is a text file with a ".c" suffix as in "test.c". This can be compiled (using *gcc*) to get an executable file:

gcc –o test test.c

The above command generates an executable file named **test** that can be invoked as follows:

./test

Question: Why do we need to use "./test" instead of just "test"?

Question: Which part of the program gets executed when invoked as above?

We may write a program that requires one or more of its inputs to be fed at the command line at the time of invocation. This may also be required when we want the program to be configured/customized based on inputs given at the time of invocation. This can be achieved by using command line arguments in C.

When the "main" procedure in a C program gets invoked it can be passed some arguments – referred to as "command line arguments". For this purpose C allows the main procedure to be defined with two parameters (whose types are fixed):

```
int main(int argc, char *argv[])
{
// body of the procedure goes here.
}
```

Note that the first parameter (argc) is an integer value that keeps a count of the number of command line arguments and the second parameter (argv) is an array of strings (i.e. values of type char *) that holds all the arguments passed via the command line to **this** program (that caused the invocation of **this** main procedure). In particular, argv[0] holds the name with which the program was invoked and for i > 0, each argv[i] holds a string corresponding to the ith word (i.e. a string separated by space) passed as the command line argument.

For instance, given the following main program:

```
/* File: test.c */
#include <stdio.h>
int main(int argc, char *argv[])
{
        printf("\n%s", argv[0]);
}
```

executing these commands
gcc –o test test.c
./test
will output
test

Exercise 1.a: If you invoke the program test as follows, what will be the value of argc?
./test hello world

Exercise 1.b: Modify the program test so that it prints "test hello world" for the above invocation. Generalize the program so that it prints the name of the program and all the command line arguments in sequence.

Exercise 1.c: Modify the program so that if a command line argument is an integer the parameter string gets converted to an integer. Refer to man pages for conversion function atoi.

2. Files and Input/Output [Expected Time: 45 minutes]

A file is an abstraction of the way data gets stored in external or secondary memory (i.e. hard disk, flash memory device, optical disk etc.). A file can be treated as a sequential access FIFO list by a program i.e. if the contents of a file were (say, for instance):
Q W E R T Y
the first accessible element is Q, then W, then E, and so on – i.e. to access E one has to access Q, then W, and then access E; and when say a value U is added to the above file, the contents would be
Q W E R T Y U
i.e. if and when a new element is added it is added to the end (of the list i.e. file).
Linux supports text and binary files. We will deal with text files for now. Text files can be accessed character by character or word by word (i.e strings separated by space). C provides an I/O library stdio that contains procedures for I/O access. The header file stdio.h contains the headers (i.e. dummy definitions) of these procedures.

Exercise 2.a: Locate the file stdio.h in your computer, read and understand at least one pair of I/O procedures. Note that the command find may be used to locate files by name. Refer to the man pages for information on how find works.

Exercise2.b: C libraries support two procedures fscanf and fprintf for reading and writing to a file. Refer to the man pages for information on how to use these procedures. They are similar to **scanf** and **printf** but take an additional (first) argument that is a file pointer.

Question: How do you obtain a file pointer?
Since files are abstractions of physical persistent storage, typically initialization and finalization are required i.e. initialization must be done before any read/write operations, and finalization must be done after all read/write operations (particularly before close of program execution).

Exercise 2.c: C libraries provide procedures fopen and fclose for initialization and finalization of a file. Refer to the man pages for information on how to use these procedures.

The typical structure of a program fragment that reads from and/or writes to a file is as follows:

FILE *f = fopen("testfile.txt"); // fopen returns a fle pointer

/* read or write operations using the file pointer f */

fclose(f);

Exercise 2.d: Write a C program that copies the contents of a file into a different file (given two filenames as command line arguments). Error-proof your argument: i.e. detect and print messages when errors occur – for instance, *file is not present, unable to read/write*, etc. Also ensure that your program terminates gracefully when errors occur.

Exercise 2.e: Modify your program (for 2d) so that if only one file name is passed as command line argument then the contents are copied to display. Note that **stdout** is a file pointer that typically refers to the output file abstraction corresponding to the display device.

3. Linked Lists [Expected Time: 75 minutes]

Exercise 3.a: Write a C program that reads the contents of a file (containing a sequence of characters or a string) into a LIFO list. Implement this list as a linked list. Note that performing insertions and deletions at the beginning of a linked list results in a LIFO list. Find out whether the character string stored in linked list is palindrome or not. To accomplish the task, copy the content of LIFO list into another linked list and check for palindrome by comparing content of each node in both the lists. Pass the filenames as command line arguments.

Exercise 3.b: Write a C program that reads the content of two different files in two separate linked lists. Both the files contain sequence of integer numbers separated by space in ascending order. Write a function which will traverse these two lists and merge them without disturbing the order. Merged list should be then copied to an output file. Pass file names as command line arguments.

Exercise:

Write a program to read a list of students of a course (student id, student name, the marks obtained by them in the course) stored in a file as input, sort the list of students on the basis of total marks stored (if marks are equal, sort by ID), and print the rank and name of the students on a comma-separated file. (rank=1 for the student having the highest total mark.)

****