60-141 – Introduction to Programming II Winter, 2014
Assignment 5

NOTE: For this assignment you will submit 2 script files. You will be writing two separate C programs that are closely related as discussed below.

Part A: File Manipulation:

In this assignment you are asked to develop a C program that maintains a sample address book system stored in a sequential text file. Start by designing a contact structure. A contact record describes the first name (char [30]), last name (char [30]), address (char [100]), Postal Code (char [6]) and phone number (char [10]).

Your program should be able to interactively ask the user for input, and support up to MAX (10) records. Next you should save the records to a sequential text file named "contactlistA.dat" (for part B "contactlistB.dat"). In this file you will store all the records in a format that you would be able to read back later when requested and display the contents accordingly. A sample run of the program, with its interactive menu is shown below.

Example:
*** Personal Contact Book v1.0 ***
1. Add new contact
2. Display current contacts
3. Search for a contact
4. Save contacts to file
5. Load contacts from file
6. Exit
> 1

Adding new contact:
First name : john
Last name : doe
Address : 123 any street
Postal code: N8N8N8
Phone : 5192533000

Add another contact? y

Adding new contact:
First name : Mary
Last name : jane
Address : 123 Park street
Postal code: X8X4X4
Phone : 5192534000
Add another contact? n

1. Add new contact
2. Display current contacts
3. Search for a contact
4. Save contacts to file

5. Load contacts from file
6. Exit
> 4
Contact List saved successfully.

1. Add new contact
2. Display current contacts
3. Search for a contact
4. Save contacts to file
5. Load contacts from file
6. Exit
> 3

What is the contact's last name? jane
Found 1 record(s):
First name : Mary
Last name : Jane
Address : 123 Park Street
Postal code: X8X-4X4
Phone : (519) 253-4000

1. Add new contact
2. Display current contacts
3. Search for a contact
4. Save contacts to file
5. Load contacts from file
6. Exit
> 6
Save your contacts to file before leaving? y
Contact List saved successfully.
Bye!

Additional Rules:

1. Your program interactively requests value from the user. Keep interacting with the user until "exit" is selected from the menu.

2. The phone number is entered in the format aaapppssss where a=area code, p=prefix, and s=suffix; and when output to the screen it should be displayed in a readable format (aaa) ppp-ssss. Note that in the file it should be stored as originally entered. (Example: in: 5192533000 out: (519) 253-3000. Consider writing to format the phone output and a function to check a valid phone input (FormatPhoneNumber and isValidPhoneNumber). Do a similar rule for the postal code to display it in the form N8N-8N8.

3. Use the function WordCap() which modifies the first character of the word (or sentence) into upper case and converts the rest of the characters into lower case.

4. Use a simple array of structures to store the records in (program) memory.

5. Use a sequential text file to store your records in the disk. Note: storing spaces especially in the address field may be problematic when attempting to read the record back. There are many ways around this (research it!), one way is to convert spaces to a special symbol, say underscore '_', right before saving it to file (EncodeSpace), and when reading the encoded_string back from the file we can convert the underscores back to spaces before we display them to the screen (DecodeSpace).

6. Implement a linear search that is capable of searching your entire array based on the last name (case insensitive search). It should display all occurrences of records with that last name with the number of records found.

7. Write any functions you feel are necessary and declare any arrays you may need. Document your technique properly.

8. To clarify, when the user is entering data for a new record from the keyboard, you should allow the user to enter spaces such as "123 any street", you then convert the string using EncodeSpace and store the new one (without the spaces/but using underscores) into the file just before writing it.

9. For simplicity, when a file is saved, the contents of the existing files are wiped out and replaced. When a file is loaded, the contents of the array in memory is wiped out and loaded with the new values.

10. Consider using a typedef for your contact structure to simplify your code a little.

Part B. Dynamic Linked Lists

Part B is based on part A with one important change in the requirements: Instead of using a single dimensional array to store the contacts, use a dynamic linked list built using a self-referential structure.

Why a dynamic linked list?

When using an array, we were required to specify its size, a fixed constant, at declaration time. Consequently, if we choose an array of size MAX, and as we present the menu to the user during program run time, we are faced with a dilemma: Either the user will enter too many records, even more than what the array could hold, and run out of storage room in the array and crash. Or, the user may not enter any records, or enter very little records, so that the memory allocated for the array is left unused and mostly to waste. Wasting primary storage memory is not good programming.

A possible solution is to allocate and use memory as you need it. For instance, if the user presses 1 to add a new record, then immediately (at run time, hence dynamically) a new memory allocation is requested and the new record is stored there. If another record is requested, then yet another memory allocation is made and "linked" to first one to form continuity in a "linked list".

You can use functions from part A (modified as needed) in part B.

REQUIREMENTS:
- Write and document a complete C program that is capable of satisfying the requirements of this assignment problem.
- UNDOCUMENTED OR IMPROPERLY DOCUMENTED code will automatically lose 50% marks.

- PLAGIARIZED work will not be graded and receive a mark of ZERO and reported according to the Senate bylaws.
- Write and document two complete C programs (Assign5a.c and Assign5b.c) that are capable of satisfying the program requirements.
- TO SUBMIT: No later than the submission deadline, your email must be received by: cs14101@uwindsor.ca, late submissions are not accepted and will receive a mark of ZERO.
- Email your work as an attachment, include both the source file (assign1.c) and the script file (assign1.txt)
- see below how to create the script file.

To create a script file (one that logs your compilation steps and your output in a text file):
1. script Assig5a4.txt
2. cat Assig5a2.c
3. gcc Assign5a.c
4. a.out
5. ls -l
6. exit (DO NOT FORGET THIS STEP!!)

Follow same steps for Assign5b.c.

Email all four files with the mail Subject field format "**Ass #5 section 51**" (replacing 51 by your actual registered lab section number) to:

cs141@courses.cs.uwindsor.ca

NOTE: Submissions that are not received correctly by the deadline will automatically receive a ZERO mark. In the event that more than one email submission is sent, only the last one (according to the date and time stamp) will be marked.

It is your responsibility to ensure the email attachment is sent correctly (readable) and to the right mailbox by the deadline. If you omit the email subject header or fail to follow the format provided above for the attachment file, your assignment may not be graded.

Late assignment submissions are not accepted!

NOTES:

1. Your assignment must be RECEIVED by the due date and time. Late assignment submissions are NOT accepted. Keep your script file, and all your code unmodified as proof of its completion in case it is not received.

2. It is your responsibility to get an early start on the assignment, research and ask questions ahead of time from the due date.

3. You must use your uwindsor email account to submit your work. Please do not use other email accounts (hotmail, yahoo etc...) for the purpose of assignment submissions.

4. Marks will be deducted for unclear code. (improper spacing and alignment, hard to read programs and missing outputs).

5. Make sure you turn in a complete script file that clearly shows: your code, your compilation process, a listing of the directory showing your source file(s) and the a.out with the date/time stamps, and the output.

6. **PLAGIARISM:** CHEATING IS NOT TOLERATED. You must submit your own work. Students who are suspected of copying someone else's work will be reported to the department's chair and the Dean of Science and be dealt with in accordance with the University policies. You should not share your code with others. Codes that are similar to each other will BOTH be reported as potential evidence of copying. It is imperative that you write your own code.

7. Authorized/limited help on this assignment may be provided directly from your Lecture or Lab instructors and Teaching Assistants.