Indian Institute of Technology Roorkee

Department of Computer Science and Engineering

CSN-261: Data Structures Laboratory (Autumn 2019-2020)

Lab Assignment-1 (L1) **Date: July 17, 2019** **Duration: 1 Week**

General Instructions:

1. Every Lab Assignment will be performed by the students individually. No group formation is required and the evaluations will be done every week for the students individually.
2. Use of Linux OS is mandatory for this Lab to complete all the Lab Assignments.
3. The total execution time or CPU time is to be reported for all the programs in each Lab Assignment.
4. The student should use Deoxygen tool (http://www.doxygen.nl) for getting automatic documentation of the codes written by him/her.
5. For version control of the written codes, the students are being instructed to learn and use any open-source CSV tool (https://www.nongnu.org/cvs) or GitHub (https://github.com).

Submission and Evaluation Instructions:

1. **Submit your** zipped folder (**<filename>.zip** or **<filename>.tar.gz**) through your account in Moodle through the submission link for this Lab Assignment in Moodle course site: https://moodle.iitr.ac.in/course/view.php?id=46
2. **Hard deadline for Final submission in Moodle: July 24, 2019 (1:00 pm Indian Time).** For any submission after Final Deadline, 20% marks will be deducted (irrespective of it is delayed by few seconds or few days). The key to success is starting early. You can always take a break, if you finish early.
3. The submitted zipped folder (**<filename>.zip** or **<filename>.tar.gz**) must contain the following:
   (a) The source code files in a folder
   (b) A report file (**<filename>.DOC** or **<filename>.PDF**) should contain the details like:
      i. Title page with details of the student
      ii. Problem statements
      iii. Algorithms and data structures used in the implementation
      iv. Snapshots of running the codes for each problems
4. The submission by each student will be checked with others' submission to identify any copy case (using such detection software). If we detect that the code submitted by a student is a copy (partially or fully) of other's code, then the total marks obtained by one student will be divided by the total number of students sharing the same code.

Instructions for L1:

1. Objective of this Lab Assignment L1 is to make the students familiar with different data structures while coding the programs in C language to solve some real-life problems.
2. The students are expected to have the basic knowledge of data structures and C programming language.
3. It is mandatory to learn and use gdb (GNU Debugger) for debugging the programs in Linux platform after installing gdb (https://www.gnu.org/software/gdb). There is also an interesting online GDB tool to learn: https://www.onlinegdb.com
4. The student will have to demonstrate and explain the coding done for this Lab Assignment L1 in the next laboratory class to be held on **July 24, 2019** for evaluation.

Problem Statement 1:

Write a C program to create a student management system, where the students' information are stored in a doubly circular linked list, as shown in Figure 1. The structure of each node from the list is shown in Figure 2. Initially, the circular doubly linked list is empty and the student personal data is entered from the filename "***StudentData.xlsx***" that contains the data of 13 students (name, D.O.B., address and phone no) in tabular form. The ***StudentData.xlsx*** file can be converted into a CSV file using Libreoffice or into any other file format readable from your C program. The program should have the following operations: **insert**, **delete**, **search**, **modify**, **sort** and **print**. While inserting, a unique roll number in the linked list is assigned to each student, where the starting roll number should be 101 and the list should always be in sorted according to their roll number (ascending order). However, when a deletion operation is performed, the roll number of the deleted student node is stored in a queue named ***unusedRollNo.*** These deleted roll numbers from the ***unusedRollNo*** queue will be allotted to the new students on next insertion operations.

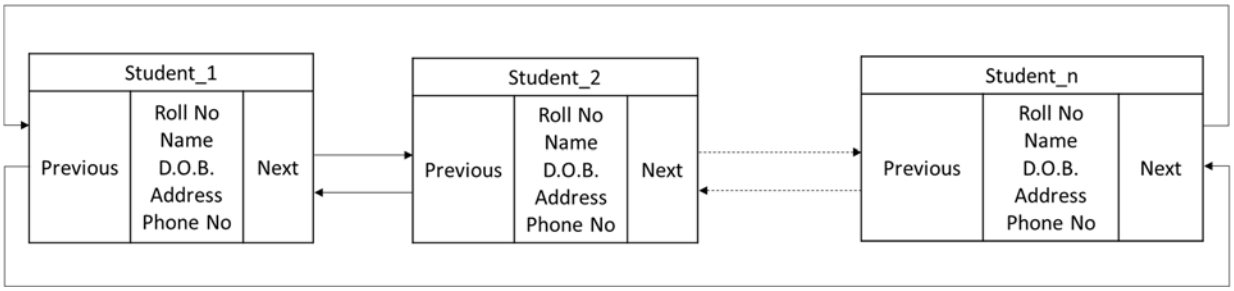

Figure 1: Doubly linked list for student data.

Student_1		
Previous Pointer	Roll No <int> Name <string> D.O.B. <string> Address <string> Phone No <int>	Next Pointer

Figure 2: Student Node

Perform the testing of your code with the following TestCases:

(Initially the list is empty)

1. delete (roll number 108) - delete the student node with roll number 108
2. insert - insert first student data from the "***StudentData.xlsx***" file (Row2)
3. insert - insert second student data from the "***StudentData.xlsx***" file (Row3)
4. insert - insert 3rd student data from the "***StudentData.xlsx***" file (Row4)
5. insert - insert 4th student data from the "***StudentData.xlsx***" file (Row5)
6. delete (roll number 102) - delete the student node with roll number 102
7. delete (roll number 101) - delete the student node with roll number 101
8. insert - insert 5th student data from the "***StudentData.xlsx***" file (Row6)
9. insert - insert 6th student data from the "***StudentData.xlsx***" file (Row7)
10. insert - insert 7th student data from the "***StudentData.xlsx***" file (Row8)
11. print - print the linked list with the roll number, name and D.O.B
12. sort (name) - sort the name according to student names
13. print - print the linked list with the roll number, name and D.O.B
14. modify (roll number 103) - modify the student node having roll number 103
15. print - print the linked list with all the records for each student

Note: In 'modify' function, the programmer can update the other fields except the roll number of a student.

Problem Statement 2:

Write a C Program for resizeable deque using dynamic memory allocation, where a deque can perform the insertion and deletion operations at its both ends. The capacity of the deque depends on the number of elements currently stored in it, according to the following two rules:

- If an element is being inserted into a deque, when it is already full, then its capacity is doubled of its current size.

- After removing an element from a deque, if the number of elements are equal to half of the capacity of the deque, then its capacity is made half of its current size.

The program should have the following three functions: **insert()**, **delete()** and **print()**.
The function **print()** should display the current size of the deque (capacity of deque) in terms of number of bytes.

Problem Statement 3:

Given three 2D arrays (for red, green and blue color pixels) of a digital image. For a particular image pixel, the color shade of that pixel is Red if the pixel value at that position of the matrix corresponding to RED is greater than that of GREEN and BLUE. Same goes for GREEN and BLUE shades also. Write a C program that can perform following operations on the given image file:

- Remove all Red shades.
- Remove all Green shades.
- Remove all Blue shades.
- RedOnly: Preserve any red shades in the image, but remove all green and blue.
- GreenOnly: Preserve any green shades in the image, but remove all red and blue.
- BlueOnly: Preserve any blue shades in the image, but remove all red and green.

Write a function **pixelValue()** that has x and y as two parameters and displays the current pixel (RED, GREEN and BLUE) values of the input image at the point with coordinates (x, y), where x and y are the row and column numbers in that image file, respectively.

Perform the testing of your code with the following TestCases:

Input: Q3_ip_Red.dat, Q3_ip_Green.dat and Q3_ip_Blue.dat are the three files with red, green and blue pixel values for the image file Q3_ip.jpg

Output after removing only green: Q3_op_Red.dat, Q3_op_Green.dat and Q3_op_Blue.dat are the output files w.r.t. red, green and blue pixel values after removing green from the input file Q3_ip.jpg, respectively.