BPDC, Dubai – II Semester, 2020-2021

Course No: CS F211 **LAB EXERCISE 5** **Course Title: DSA**

Date: March 02 2021

Implement subroutines that deploys the basic operations of a linear singly linked list. You are provided with a program listWrapper.c. The program has 5 functions createNode(), insertNode(),searchNode(),deleteNode(), and displayList(), where you are supposed to implement the last 3 of those. The prototype and other details for each of those functions are encoded as comments within the program itself.

Further, you have been provided 2 additional files, testCaseForList.txt and output.txt. The program reads the input from the first file and the second one is for your reference. Before termination, the program makes a call to the function diplayList(), and the output should exactly match with the content of the file output.txt. In addition, the program generates an additional file testResult.txt which summarizes the sequence of operations that the program performs on the list.

(a) Write a C/C++/Java program to implement a stack using linear singly linked list. Your program should read input from the file testcaseForStack.txt. Each line in the file begins with either 1 or 2. 1 stands for push and 2 stands for pop. Those lines beginning with 1 further contains the details of the record you are supposed to push. The record structure remains the same as the student record in the first part. To be specific, you are supposed to implement the following functions.

struct node*pop(struct node**top)//removes the topmost node from the stack and returns the address of the same, returns NULL if in case the stack is empty

int isEmpty()//returns 1 if the stack is empty (top=NULL) and 0 otherwise

void push(struct node**top,struct node*temp)//temp is the address of the node that you are supposed to push. Note that the stack never gets full and hence push never fails.

Note that you have to create node using malloc function, fill in data that you read from the file and then invoke push. On each successful pop, print the data associated with the node that you popped from your main function and free up the corresponding node memory (print an error message if in case the stack was empty). In addition, make sure that you declare top as a local variable in your main function and pass reference to top variable (&top) both to push and pop functions so as to update the top variable from within the function if required.