ALGORITHMS & DATA STRUCTURES
L.Ouarbya

TOPIC 1: LINEAR DATA STRUCTURES
LAB. SUBMISSION

In this activity, you will work **individually** implementing different functions for a linked list. Remember that you can discuss ideas and questions with your classmates, but you cannot see other's work neither others can see your individual work.

After finishing the implementation of your functions, you must answer a Lab Submission Quiz that will test your knowledge on the understanding of the code you developed.

WEEK 1: Linked Lists

Download the skeleton code[1] from learn.gold (Term 2- Week 1) and do the activities described below.

Notice that these activities consider that you work 10 hours a week for this module: This includes lecture time as well watching videos and understanding the concepts presented in them.

Activity 1: Have a good read at the code and identify the main blocks:
- how a node is defined
- how the head of the list is defined
- how the insertion of a new node at the beginning of the list is done
- how the content of the list is printed
- how the element x (its first occurrence) is deleted from the list
- how the main function allows you to interact with the user

Activity 2: Execute the code and make sure you know how to use it (at user level).

Activity 3: Implement the function lenghtList() that returns the number of elements of the list. Modify the main menu to include this function in the options.

Activity 4: Implement the function printListK(K) that prints the first K elements of the list. The number K is entered by the user. Modify the main menu to include this function in the options. If the list has a number of items N lower than K, the message "The list

[1] C programmers can select one of two options: having a single .cpp file or having multiple files (headers and .cpp files). The second option is the most used in professional environments, as it helps organise the information in simpler files. At this stage, you can choose any of them.

has only N elements" must be displayed. If the user enters a negative number, an error message must be printed.

Activity 5: Implement the function countList(x) that returns the number of times x appears in the list. The number x is entered by the user. Modify the main menu to include this function in the options.

Activity 6: Implement the function searchList(x) that returns TRUE if number x is in the list and FALSE if not. The number x is entered by the user. Modify the main menu to include this function in the options.

Activity 7: Implement the function insert_end(x) that inserts number x at the end of the list. The number x is entered by the user. Modify the main menu to include this function in the options.

Activity 8: Implement the function clear() that deletes all element in the list. Modify the main menu to include this function in the options.

Activity 9: Implement the function max() that returns the value of the maximum element in the list. Modify the main menu to include this function in the options.

Activity 10: Implement the function delete_first() that returns the value of the first element in the list and then, delete it from the list. Modify the main menu to include this function in the options.

Activity 11: Calculate the time complexity of every function implemented (Activity 3-10). Use Theta notation for this.

WEEK 2: Stacks & Queues

Implement a programme that emulates the operation of a stack and another programme that emulates the operation of a queue. Both programmes **must use** linked lists.

The stack data structure should have the following functions implemented:

- POP(), it removes the last element inserted into the stack
- PUSH(x), it inserts number x into the stack
- PEEK(), it returns the last element inserted into the stack (without removing it from the stack)
- ISEMPTY(), it returns TRUE if the stack is empty. False, otherwise.

Your programme for the stack should have a user menu as follows:

Please, enter your option:

0. Quit the programme
1. PUSH(x)
2. POP
3. PEEK
4. ISEMPTY

The queue data structure should have the following functions implemented:

- QUEUE(x), it inserts element x at the end of the queue
- DEQUEUE(), it removes the first element in the queue
- PEEK(), it returns the value of the first element in the queue
- ISEMPTY(), it returns TRUE if the queue is empty. False, otherwise.

Your programme for the queue should have a user menu as follows:

Please, enter your option:

0. Quit the programme
1. ENQUEUE(x)
2. DEQUEUE
3. PEEK
4. ISEMPTY