UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS110 - Program design: Introduction

Reward Practical: Recursive Link List

Release date: 31-10-2022 at 06:00
Due date: 11-11-2022 at 23:59
Total Marks: 61

Contents

1 General instructions:

- This assignment should be completed individually, no group effort is allowed.
- Be ready to upload your assignment well before the deadline as no extension will be granted.
- You may not import any of C++'s built-in data structures. Doing so will result in a mark of zero. You may only make use of 1-dimensional native arrays where applicable. If you require additional data structures, you will have to implement them yourself.
- If your code does not compile you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- If your code experience a runtime error you will be awarded a mark of zero. Runtime errors are considered as unsafe programming.
- All submissions will be checked for plagiarism.
- Read the entire specification before you start coding.
- Ensure your code compiles with C++98
- You may not use any loop structure except recursion. The following are examples of "illegal" loops:
  - For loops
  - Jumps
  - While loops
  - Do while loops

2 Plagiarism

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). **If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding.** Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

3 Outcomes

The aim of this practical is to possibly reward students that put in the extra effort this semester. This practical will focus on link lists and recursion in an attempt to gain familiarity with the concepts in preparation for COS212, COS214, COS341, COS333 and WTW285.

4 Introduction

Implement the UML diagram and functions as described on the following pages.

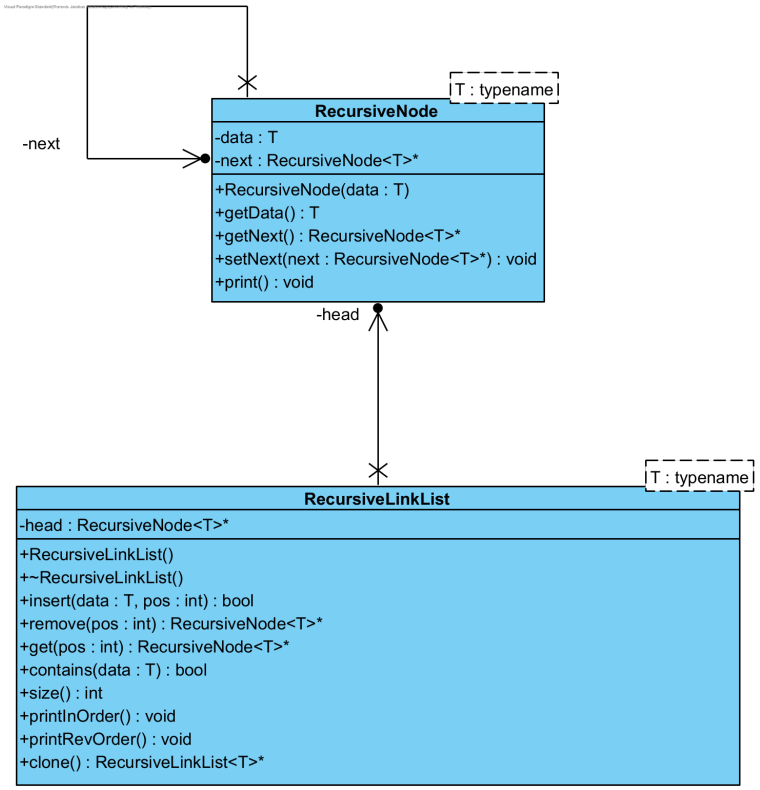

Figure 1: Class diagrams

You have been provided with the header files for all the above classes.
Please note that in the provided RecursiveLinkList header there are a number of private functions. These are there to guide you in case you are unsure where to start.

5 Classes:

5.1 RecursiveNode

- Members:
  - next: RecursiveNode<T>*
    - This is a pointer to the next recursive node.
  - data: T
    - This is the data contained in the current node.
- Functions:
  - RecursiveNode(data: T)
    - This is the constructor for the RecursiveNode class.
    - Initialize the data member variable with the passed in parameter.
    - Initialize next to NULL.
  - getData() const: T
    - This function should return the data member variable.
  - getNext() const: RecursiveNode<T>*
    - This function should return the next member variable.
  - setNext(next: RecusriveNode<T>*): void
    - This function should set the next member variable to the passed in parameter.
  - print(): void
    - This function has been provided and should not be altered.

5.2 RecursiveLinkList

Please note that in the provided RecursiveLinkList header there are a number of private functions. These are there to guide you in case you are unsure where to start.
Please note you may change the private functions but not the public functions.

- Members:
  - head: RecursiveNode<T>*
    - This is the head node for the list.
- Functions:
  - RecursiveLinkList()
    - This is the constructor for the class.
    - This function should initialize head to NULL
  - ∼ RecursiveLinkList()
    - This is the destructor for the class.
    - This function should deallocate all the nodes in the list.
  - insert(data: T, pos: int): bool
    - This function should add the data to the list at the passed in position.
    - If the passed in position is negative then the function should return false;

    - If the passed in position is bigger then the size of the list then the data should be added to the rear of the list.
    - If the passed in position is 0 then the data should be added to the front of the list.
    - If the data was added to the list return true.
  - remove(pos: int): RecursiveNode<T>*
    - This function should remove the node at the given position and return it.
    - If the node was not able to be removed the function should return NULL.
    - If the passed in position is invalid the function should return NULL.
  - get(pos: int): RecursiveNode<T>*
    - This function should return the node at the given position.
    - If the position is invalid or greater then the size of the list the function should return NULL.
  - contains(data: T): bool
    - This function should return true if the passed in data is contained in the list and false otherwise.
  - size(): int
    - This function should return the size of the list.
    - If head is NULL the function should return 0.
  - printInOrder(): void
    - This function should print the values in the order of the link list from head to the last node.
    - Use the provided print function of the RecursiveNode class to print each node.
  - printRevOrder(): void
    - This function should print the values in the reverse order of the link list from the last node to the head.
    - Use the provided print function of the RecursiveNode class to print each node.
  - clone(): RecursiveLinkList<T>*
    - This function should return a deep copy of the current link lists.

6 Source files

Please upload the following files.

- RecursiveLinkList.{h,cpp}
- RecursiveNode.{h, cpp}

7 Allowed libraries

- cstddef
  - Note this is imported such that the NULL constant is defined.
- iostream
  - For the print function.

8 Helper functions

Note for this practical you **may** add your own helper functions to the RecursiveLinkList or RecursiveNode classes.

9 Restriction

Note as stated earlier you may only use recursive loops for this practical. The use of any iterative or jump loops will result in a mark of 0. Also note do not use the following words as comments in your code:

- for
- while
- goto
- do

10 Note on warnings:

In this practical you may see the following or similar warnings:

```
warning: converting to non-pointer type 'double' from NULL [-Wconversion-NULL]
```

This is just due to conversions between primative data types and NULL constant. These warnings should not effect your codes performance on FitchFork,

11 Submission

You need to submit your source files, only the cpp files, on the Fitch Fork website (https://ff.cs.up.ac.za/). All methods need to be implemented (or at least stubbed) before submission. Place the above mentioned files in a zip named uXXXXXXXX.zip where XXXXXXXX is your student number. Your code should be able to be compiled with the C++98 standard
For this practical you will have 1 upload opportunities. Upload your archive to the Reward Practical slot on the Fitch Fork website.