Fall 2012 BIOSTAT 615/815 Problem Set #2

Due is Saturday October 6th, 2012 11:59PM by google document (shared to hmkang@umich.edu and atks@umich.edu) containing the source code and answers to the questions. Also email of the compressed tar.gz file containing all the source codes.

Problem 1. A new sorting algorithm (50 pts)

Let’s define a MYSORT algorithm as follows.

**Data:** An unsorted list \( A[1 \ldots n] \)

**Result:** The list \( A[1 \ldots n] \) is sorted

\[
\text{for } i = 1 \text{ to } n \text{ do}
\]

\[
\text{for } j = i + 1 \text{ to } n \text{ do}
\]

\[
\text{if } A[i] > A[j] \text{ then}
\]

\[
\text{EXCHANGE}(A[i], A[j]);
\]

\[
\text{end}
\]

\[
\text{end}
\]

(a) (15pts) Prove that the MYSORT algorithm works correctly.

(b) (10pts) What is the time complexity of MYSORT algorithm? Explain briefly.

(c) (25pts) Implement a program mySort.cpp by modifying insertionSort.cpp.

In your google document, in addition to answer to (a) and (b), include full source code. Your source code file name must be hw-2-1.cpp. Do not include any other files in your .tar.gz submission.

Problem 2 - Implementing doubly linked list (50 pts)

Implement a doubly linked list, following the skeleton below. hw-2-2.cpp is already implemented below

```cpp
#include <iostream>
#include "doublyLinkedList.h"

int main(int argc, char** argv) {
    std::string cmd;
    int input;
    doublyLinkedList<int> list;
    std::cout << "Type [s/i/d] [value] : ";
    while( std::cin >> cmd >> input ) {
        if ( cmd == "s" ) {
            std::cout << "Searching " << input << " from the list : returning " << list.search(input) << ". ";
        } else if ( cmd == "i" ) {
            std::cout << "Inserting " << input << " into the list. ";
            list.insert(input);
        } else if ( cmd == "d" ) {
            std::cout << "Deleting " << input << " from the list : returning " << list.remove(input) << ". ";
        } else {
            std::cerr << "ERROR: Cannot recognize the command " << cmd << std::endl;
        }
        std::cout << "Current list is ";
        list.print();
        std::cout << std::endl << std::endl << "Type [s/i/d] [value] : ";
    }
    return 0;
}```
doublyLinkedList.h

#ifndef __DOUBLY_LINKED_LIST_H // This is useful to avoid redundant inclusion
#define __DOUBLY_LINKED_LIST_H // to avoid redundant inclusion

#include <iostream>
#include "doublyLinkedListNode.h"

template <class T>
class doublyLinkedList {
    protected:
        doublyLinkedListNode<T>* head;
        doublyLinkedList(doublyLinkedList& a) {}; // prevent copying

t    public:
        doublyLinkedList();
        ~doublyLinkedList();
        void insert(const T& x);
        bool search(const T& x);
        bool remove(const T& x);
        void print();
    }

    /*** YOU NEED TO DEFINE THE MEMBER FUNCTIONS HERE ***/

#endif

doublyLinkedListNode.h

#ifndef __DOUBLY_LINKED_LIST_NODE_H // This is useful to avoid redundant inclusion
#define __DOUBLY_LINKED_LIST_NODE_H // to avoid redundant inclusion

#include <iostream>

template <class T>
class doublyLinkedListNode {
    protected:
        doublyLinkedListNode<T>* prev;
        T value;
        doublyLinkedListNode<T>* next;

        doublyLinkedListNode(doublyLinkedListNode<T>* p, const T& x, doublyLinkedListNode<T>* n);
        ~doublyLinkedListNode();

        bool search(const T& x);
        doublyLinkedListNode<T>* remove(const T& x);
        void print();

        template <class S> friend class doublyLinkedList;
    }

    /*** YOU NEED TO DEFINE THE MEMBER FUNCTIONS HERE ***/

#endif

Below is the expected output of an example run.

user@host:~/Private/biostat615/hw2$ ./hw-2-2
Type [s/i/d] [value] : i 1
Inserting 1 into the list. Current list is (1)
Type [s/i/d] [value] : i 10
Inserting 10 into the list. Current list is (10,1)

Type [s/i/d] [value] : i 2
Inserting 2 into the list. Current list is (2,10,1)

Type [s/i/d] [value] : s 10
Searching 10 from the list : returning 1. Current list is (2,10,1)

Type [s/i/d] [value] : d 10
Deleting 10 from the list : returning 1. Current list is (2,1)

Type [s/i/d] [value] : s 10
Searching 10 from the list : returning 0. Current list is (2,1)

Type [s/i/d] [value] : d 2
Deleting 2 from the list : returning 1. Current list is (1)

Type [s/i/d] [value] : d 1
Deleting 1 from the list : returning 1. Current list is (EMPTY LIST)

Note the following requirements.

- Beware of memory leak. Make sure that the number of objects created by `new` matches the number of deleted objects. You may insert a small debug code to count the number of constructor and destructor calls.

- Your implementation needs to behave correctly for any input sequence.

- If your implementation is unreasonably inefficient, you may not obtain a full credit.

In your google document, include full source code (brief comments would be helpful) and an example output. Your source code names must be `hw-2-2.cpp`, `doublyLinkedList.h`, `doublyLinkedListNode.h`. Do not include any other files in your .tar.gz submission.

Problem 3 (BIOSTAT815 only) - Binary search tree with parents (50 pts)

Modify `myTree.h` and `myTreeNode.h` from lecture 7 by adding pointer to its parent node, so that `myTreeNode` contains a member `parent`, in addition to `left` and `right`. Modify the implementations of the class accordingly. Make `hw-2-3.cpp` as a copy of `hw-2-2.cpp`, by substituting `doublyLinkedList` into the modified version of `myTree`. The tree must behave correctly with a reasonable efficiency.

In your google document, include full source code and an example output. Your source code names must be `hw-2-3.cpp`, `myTree.h`, `myTreeNode.h`. Do not include any other files in your .tar.gz submission.