LASALLE COLLEGE VANCOUVER

Templates & Introduction to STL: Vectors

VGP 131 - Object Oriented Programming in C++ II

Instructor: Ivaldo Tributino

April 14, 2022

ASSIGNMENT INSTRUCTION

- The assignment must be submitted by April 24, 2022.
- Each problem presents its own score, the sum of all scores is 100.
- The LinkedList class can be found in the appendix.

Student's Number:
Student's Name:

(20 POINTS) PROBLEM 1

Implement the function insertPosition of the template class linkedList.

```
// Insert data at a specific index in the list.
void insertPosition(unsigned index, const T & data)
```

(20 POINTS) PROBLEM 2

Implement the function deleteNote of the template class linkedList.

```
// Remove all occurrences of a data in the list
void deleteNote(const T & data);
```

(20 POINTS) PROBLEM 3

Create a container, vector or list, to store the lines of the mbox-short.txt file. Then clean up the data using **container methods** and **algorithms functions**: keep only emails that are in lines starting with "From: " and eliminate duplicates.

```
287       Fri, 4 Jan 2008 20:02:46 +0000 (GMT)
288   Received: from nakamura.uits.iupui.edu (nakamura.uits.iupui.edu [134.68.220.122])
289       by shmi.uhi.ac.uk (Postfix) with ESMTP id AB4D042F4D
290       for <source@collab.sakaiproject.org>; Fri,  4 Jan 2008 20:02:50 +0000 (GMT)
291   Received: from nakamura.uits.iupui.edu (localhost [127.0.0.1])
292       by nakamura.uits.iupui.edu (8.12.11.20060308/8.12.11) with ESMTP id m04K1cXv007740
293       for <source@collab.sakaiproject.org>; Fri, 4 Jan 2008 15:01:38 -0500
294   Received: (from apache@localhost)
295       by nakamura.uits.iupui.edu (8.12.11.20060308/8.12.11/Submit) id m04K1cO0007738
296       for source@collab.sakaiproject.org; Fri, 4 Jan 2008 15:01:38 -0500
297   Date: Fri, 4 Jan 2008 15:01:38 -0500
298   X-Authentication-Warning: nakamura.uits.iupui.edu: apache set sender to zqian@umich.edu using -f
299   To: source@collab.sakaiproject.org
300   From: zqian@umich.edu
301   Subject: [sakai] svn commit: r39766 - site-manage/branches/sakai_2-4-x/site-manage-tool/tool/src/java/
302   X-Content-Type-Outer-Envelope: text/plain; charset=UTF-8
303   X-Content-Type-Message-Body: text/plain; charset=UTF-8
304   Content-Type: text/plain; charset=UTF-8
305   X-DSPAM-Result: Innocent
306   X-DSPAM-Processed: Fri Jan  4 15:03:18 2008
307   X-DSPAM-Confidence: 0.7626
308   X-DSPAM-Probability: 0.0000
309
```

Figure 0.1: mbox-short.txt file

(20 POINTS) PROBLEM 4

Define a class linkedListType to implement the basic operations on a linked list as an abstract class. Using the principle of inheritance, derive orderedLinkedfrom the class linkedListType. The class orderedLinkedList would arrange elements according to some comparison criteria, usually less than or equal to. That is, these lists will be in ascending order. (Hint: see Chapter 18 of C++ Programming From Problem Analysis to Program Design - D. S. Malik).

```
template <class Type>
struct nodeType{
   Type info;
   nodeType<Type> *link;
};

template <class Type> class linkedListIterator {
public:
   linkedListIterator();
   linkedListIterator(nodeType<Type> *ptr);
   Type operator*();
   linkedListIterator<Type> operator++();
   bool operator==(const linkedListIterator<Type>& right) const;
   bool operator!=(const linkedListIterator<Type>& right) const;
private:
   nodeType<Type> *current;
};

template <class Type> class linkedListType {
```

```
public:
   const linkedListType<Type>& operator=(const linkedListType<Type>&);
   void initializeList();
   bool isEmptyList() const;
   void print() const;
   int length() const;
   void destroyList();
   Type front() const;
   Type back() const;
   virtual bool search(const Type& searchItem) const = 0;
   virtual void insertFirst(const Type& newItem) = 0;
   virtual void insertLast(const Type& newItem) = 0;
   virtual void deleteNode(const Type& deleteItem) = 0;
   linkedListIterator<Type> begin();
   linkedListIterator<Type> end();
   linkedListType();
   linkedListType(const linkedListType<Type>& otherList);
   ~linkedListType();
protected:
   int count;
   nodeType<Type> *first;
private:
   void copyList(const linkedListType<Type>& otherList);
};

template <class Type>
class orderedLinkedList: public linkedListType<Type> {
public:
   bool search(const Type& searchItem) const;
   void insert(const Type& newItem);
   void insertFirst(const Type& newItem);
   void insertLast(const Type& newItem);
   void deleteNode(const Type& deleteItem);
};
```

(20 POINTS) PROBLEM 5

Implement a template doubly linked list with overload operator [], a display function, and at least one deleteNote and one addNote function.

```
template <class Type>
struct nodeType
{
    Type info;
    nodeType<Type> *next;
    nodeType<Type> *back;
};
```

```
template <class Type>
class doublyLinkedList {
public:
   const Type& operator[](unsigned index);
   const doublyLinkedList<Type>& operator=(const doublyLinkedList<Type> &);
   void initializeList();
   bool isEmptyList() const;
   void destroy();
   void print() const;
   void reversePrint() const;
   int length() const;
   Type front() const;
   Type back() const;
   bool search(const Type& searchItem) const;
   void insert(const Type& insertItem);
   void deleteNode(const Type& deleteItem);
   doublyLinkedList();
   doublyLinkedList(const doublyLinkedList<Type>& otherList);
   ~doublyLinkedList();
protected:
   int count;
   nodeType<Type> *first;
private:
   void copyList(const doublyLinkedList<Type>& otherList);
};
```