**EE2410 Data Structure Hw #3 (Chapter 4 Linked Lists of textbook)**

**due date 5/5/2024(Sun.) 23:59**

**Student No.: \_\_\_\_111060005\_\_\_\_\_**

**Name: \_\_\_\_\_\_\_胡昱煊\_\_\_\_\_\_\_**

**Note:** Use MS Word to **edit this file** by directly typing your student number and name in above blanks and your answer to each homework problem right in the **Sol:** blanks as shown below. Then save your file as **Hw3-SNo.pdf**, where SNo is your student number. Submit the **Hw3-SNo.pdf** file via eLearn. The grading will be based on the correctness of your answers to the problems, and the **format requirement**. Fail to comply with the aforementioned format (file name, header, problem, answer, problem, answer,…), will certainly degrade your score. If you have any questions, please feel free to ask. Submit your homework before the deadline (midnight of 4/28). Fail to comply (**late** homework) will have ZERO score. **Copy** homework will have ZERO score (both parties) and SERIOUS consequences.

1. (21%)

Given a template linked list **L** instantiated by the Chain class with a pointer **first** to the first node of the list as shown in Program 4.6 (textbook). The node is a ChainNode object consisting of a template data and link field. **Formulate an algorithm** (pseudo code OK, graph + explanation, C++ code not necessary) which will

1. count the number of nodes in L. Explain your algorithm properly (using either text or graphs).
2. change the data field of **the kth node** (the first 1st node start at index 0) of L to the value given by Y. Explain your algorithm properly (using either text or graphs).
3. perform an insertion to the **immediate** **before of the kth node** in the list L. Explain your algorithm properly (using either text or graphs).
4. **delete every other node** of L beginning with node first (i.e., the first, 3rd, 5th,…nodes of L are deleted). Explain your algorithm properly (using either text or graphs).
5. divides the given list into two sublists of (almost) equal sizes (**divideMid()**). Suppose myList points to the list with elements 34 65 27 89 12 (in this order). The statement: myList.divideMid(subList); divides myList into two sublists: myList points to the list with the elements 34 65 27, and subList points to the sublist with the elements 89 12. Formulate a step-by-step algorithm to perform this task. Explain your algorithm properly (using either text or graphs).
6. **deconcatenate** (or **split**) a linked list L into two linked list. Assume the node denoted by the pointer variable split is to be the first node in the second linked list. Formulate a step-by-step algorithm to perform this task. Explain your algorithm properly (using either text or graphs).
7. **merge** the two chains: L1 = (x1,x2,..,xn) and L2 = (y1,y2,…,ym) together to obtain the chain L3 = (x1,y1,x2,y2,…,xm,ym,xm+1,..,xn) if n>m and L3 = (x1,y1,x2,y2,…,xn,yn,yn+1,..,ym) if n<m. Explain your algorithm properly (using either text or graphs).

**Sol:**

1. (18%)

Given a **circular linked list L** instantiated by class CircularList containing a private data member, **first** pointing to the first node in the circular list as shown in Figure 4.14.



Fig. 4.14 A circular linked list

**formulate algorithms** (pseudo code OK, graph + explanation, C++ code not necessary) to

1. count the number of nodes in the circular list. Explain your algorithm properly (using either text or graphs)
2. insert a new node at the front of the list. Discuss the time complexity of your algorithm. Explain your algorithm properly (using either text or graphs)
3. insert a new node at the back (right after the last node) of the list. Discuss the time complexity of your algorithm. Explain your algorithm properly (using either text or graphs)
4. delete the first node of the list. Discuss the time complexity of your algorithm. Explain your algorithm properly (using either text or graphs)
5. delete the last node of the list. Discuss the time complexity of your algorithm. Explain your algorithm properly (using either text or graphs).
6. Repeat (a) – (e) above and (b) – (g) in Problem 1 above if the circular list is modified as shown in Figure 4.16 below by introducing a dummy node, header.

Figure 4.16 Circular list with a header node

**Sol:**

1. (10%)

Suppose we have a pointer to a node in a **singly linked list** that is guaranteed not to be the last node in the list. We do not have pointers to any other nodes (except by following links). Describe an **O(1)** algorithm that logically removes the value stored in such a node from the linked list, maintaining the integrity of the linked list.

Sol:

1. (5%)

Let x be a node in a **singly linked circular list**. Write a C++ function to delete the data in this node. Following the deletion, the number of nodes in the list is one less than before deletion. **Your function must run in O(1) time.**

Sol:

1. (5%)

One way to implement a queue is to use a circular linked list. In a circular linked list, the last node’s next pointer points at the first node. Assume the list **does not contain a header** and that we can maintain, at most, **one iterator corresponding to a node** in the list. For which of the following representations can all basic queue operations be performed in constant worst-case time? Justify your answers.

a. Maintain an iterator that corresponds to the first item in the list.

b. Maintain an iterator that corresponds to the last item in the list.

Sol:

1. (10%)

Suppose that a singly linked list is implemented with both a header and a tail node. Describe constant-time algorithms to

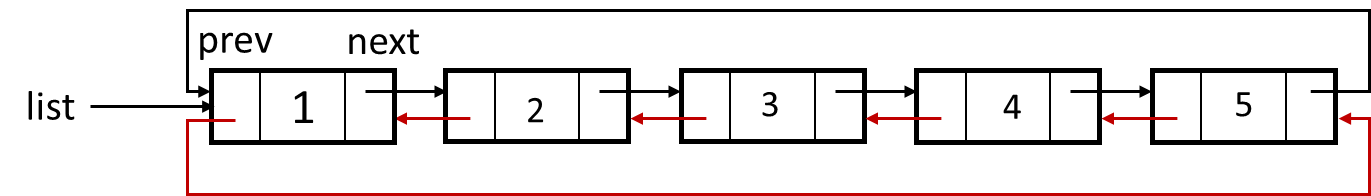
a. insert item x before position p (given by an iterator)

b. remove the item stored at position p (given by an iterator)

Sol:

1. (5%)

Assume that a circular doubly linked list has been created, as shown below. After each of the following assignments, indicate changes made in the list by showing which links have been modified. The second assignment should make changes in the list modified by the first assignment, and so on.



list->next->next->next = list->prev;

list->prev->prev->prev = list->next->next->next->prev;

list->next->next->next->prev = list->prev->prev->prev;

list->next = list->next->next;

list->next->prev->next = list->next->next->next;

Sol:

1. (6%)

Show what is produced by the following C++ code. Assume the node is in the usual data-link form with the data of type int. (list and ptr are pointers of type nodeType.)

a. list = new nodeType;

list->data = 10;

ptr = new nodeType;

ptr->data = 13;

ptr->link = NULL;

list->link = ptr;

ptr = new nodeType;

ptr->data = 18;

ptr->link = list->link;

list->link = ptr;

**cout << list->data << " " << ptr->data << " ";**

ptr = ptr->link;

**cout << ptr->data << endl;**

b. list = new nodeType;

list->data = 20;

ptr = new nodeType;

ptr->data = 28;

ptr->link = NULL;

list->link = ptr;

ptr = new nodeType;

ptr->data = 30;

ptr->link = list;

list = ptr;

ptr = new nodeType;

ptr->data = 42;

ptr->link = list->link;

list->link = ptr;

ptr = List;

while (ptr != NULL)

{

**cout << ptr->data << endl;**

ptr = ptr->link;

}

Sol:

1. (10%)

What is the output of the following program segment?

list<int> intList;

ostream\_iterator<int> screen(cout, " ");

list<int>::iterator listIt;

intList.push\_back(5);

intList.push\_front(23);

intList.push\_front(45);

intList.pop\_back();

intList.push\_back(35);

intList.push\_front(0);

intList.push\_back(50);

intList.push\_front(34);

**copy(intList.begin(), intList.end(), screen);**

cout << endl;

listIt = intList.begin();

intList.insert(listIt,76);

++listIt;

++listIt;

intList.insert(listIt,38);

intList.pop\_back();

++listIt;

++listIt;

intList.erase(listIt);

intList.push\_front(2 \* intList.back());

intList.push\_back(3 \* intList.front());

**copy(intList.begin(), intList.end(), screen);**

cout << endl;

Sol:

1. (10%)

Suppose the input is:

18 30 4 32 45 36 78 19 48 75 -999

What is the output of the following C++ code? (The class unorderedLinkedList is as is name.)

unorderedLinkedList<int> list;

unorderedLinkedList<int> copyList;

int num;

cin >> num;

while (num != -999)

{

if (num % 5 == 0 || num % 5 == 3)

list.insertFirst(num);

else

list.insertLast(num);

cin >> num;

}

**list.print();** //print out the list node data one by one.

cout << endl;

copyList = list;

copyList.deleteNode(78);

copyList.deleteNode(35);

cout << "Copy List = ";

**copyList.print();**

cout << endl;

Sol: