Mariya Eggensperger
CST 370, Spring 2017
Dr. Feiling Jia
Design / Analysis
of Algorithms

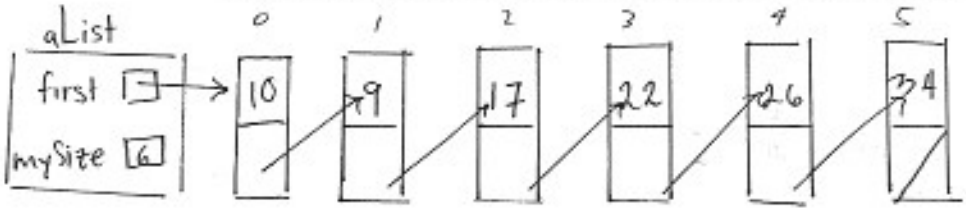
CST 370
Homework (Linked Lists)

1. Suppose that you are given a linked list as shown below. You can read about **linked lists** from section 6.4 and 6.5 of the book. Source code describing the operation of a Linked List is available on iLearn (LinkedList.h, LinkedList.cpp and Sample_LinkedList_Tester.cpp). **(30 points)**

Assume that there is a function insert (as defined below) to add a node in the linked list. Read the insert function very carefully.

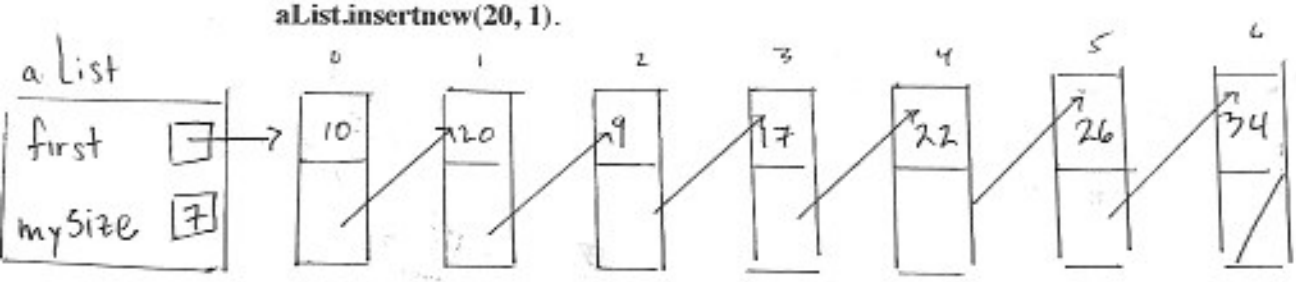
```
void List::insertnew(ElementType dataVal, int index)
{
        mySize++;
        Node * newPtr = new Node(dataVal);
        Node * predPtr = first;

        for(int i = 0; i <= index; i++)
    {
            predPtr = predPtr->next;
    }
        newPtr->next = predPtr->next;
        predPtr->next = newPtr;
}
```

(a) Draw the updated linked list after the execution of **aList.insertnew(10, 0)**.



10 dataVal
0 array Index

(b) From the result of the above question (a), draw the updated linked list after the execution of **aList.insertnew(20, 1)**.



20 dataVal
0 array Index

2. The following presents the **insertnew()** member function for a **static array-based list**. **(30 points)**

```
void List::insertnew (ElementType item, int pos)
{
   if (pos < 0 || pos > mySize) {
      cerr << "Illegal location: " << pos << "\n";
      return;
   }

   for(int i = mySize; i > pos; i--) {
      myArray[i] = myArray[i - 1];
   }

   myArray[pos] = item;
   mySize++;

   for (int i = 0; i < mySize; i++) {
      cout << myArray[i] << "  ";
   }
   cout << endl;
   return;
}
```

Assume that the following code is a part of a client program. Present the execution result. You can assume that mySize is initially 0. You can read about the **static array-based list** from section 6.2 of the book. Source code described in the book is available on the book's website (Figure 6.1).

```
List intList;

intList.insertnew(100, 1);
intList.insertnew(200, 2);   index 2
intList.insertnew(300, 0);   index 0
intList.insertnew(400, 1);
```

See next page for Answer →

2 Output

```
/Users/student/Desktop/M1_LinkedList/cmake-build-debug/M1_LinkedList

---------- LINKED LIST ----------

intList is: 300 400

intList is: 300 400 300 100 200 400

intList is: 300 400 300 100 200 400 300 100 200 400

intList is: 300 400 300 100 200 400 300 100 200 400 300 100 200 400

Type the index of the element you would like to remove.
Illegal insertion at position 1
Illegal insertion at position 2
```

3. Consider the Linked Lists files available on iLearn (LinkedList.h, LinkedList.cpp and Sample_LinkedList_Tester.cpp). Write a member function to find the mean of the values in a linked list **(40 points).**

Please see attachments →.

Mariya Eggensperger
CST 370, Spring 2017
Dr. Feiling Jia
Design/Analysis
of Algorithms

Solution to #3

LinkedList.h definitions

```
void evenMean();
/*-----------------------------------------------------------------------
Determine the mean value among the even elements in the list.
Precondition: none
Postcondition:
-----------------------------------------------------------------------*/

void oddMean();
/*-----------------------------------------------------------------------
Determine the mean value among the odd elements in the list.
Precondition: none
Postcondition:
-----------------------------------------------------------------------*/
```

LinkedList.cpp implementation

CMakeLists.txt × LinkedList.h × LinkedList.cpp × Linked_List_Tester.cpp ×

```cpp
// Definition for the mean in an even length linked list
void LinkedList::evenMean() {
    Node * fastPtr = first;
    Node * slowPtr = first;
    if (first != NULL) {
        while (fastPtr!=NULL && fastPtr->next!=NULL) {
            fastPtr = fastPtr->next->next;
            slowPtr= slowPtr->next;
        }
        cout << slowPtr->data;
    }
}
// Definition for the mean in an odd length linked list
void LinkedList::oddMean() {
    int count = 0;
    Node *middle = first;
    while(first!= NULL){
        if(count%2==1){
            middle= middle->next;

        }
        count++;
        first=first->next;
    }
    if(middle!=NULL){
        cout << middle->data;
    }
}
```

Linked_List_Tester.cpp

```
LinkedList intList;


cout << "\n\nConstructing intList" << endl << endl;

// Test insert()
intList.insert(1, 0);
intList.display(cout);
cout << endl;

intList.insert(2, 1);
intList.display(cout);
cout << endl;

intList.insert(2, 2);
intList.display(cout);
cout << endl;

intList.insert(3, 3);
intList.display(cout);
cout << endl;

cout << "\nFinal linked list: " << endl;
intList.display(cout);
cout << endl;
cout << "\nEven mean: " << endl;
intList.evenMean();
cout << endl;
cout << "\nOdd mean" << endl;
intList.oddMean();
cout << endl;
```

LinkedList OUTPUT

```
Run  Build All

Constructing intList

1
1 2
1 2 2
1 2 2 3

Final linked list:
1 2 2 3

Even mean:
2

Odd mean
2

Process finished with exit code 0
```