Quiz#3

[SLL] Singly Linked List 에 관하여 답하시오

1. 마지막 노드를 맨 앞으로 이동시키는 기능을 수행하는 MakefirstLast ADT를 작성하시오


```
Void MakefirstLast(){
 Node* p, *q;       q=p = head;
   while (p->next != 0)
       {q = p; p = p->next; }
   q->next = null;
   p->next = head;
   head=p;
}
```

2. Singly Linked List에서 맨 마지막 노드의 직전에 노드를 insert하는 <u>insert ADT</u>를 작성하시오.

ex) insert D **List before operation**: A→B→C **List after operation**: A→B→**D**→C

(sol)

```
void insert(int data){
        Node* temp = new Node(data); temp->data= 'D';
        Node *p, *q;      p = q = head;

      if (head ==0) {head = temp; return}    // if empty list
      else if (head->next ==0){                  // insert before head node
        temp->link = head; head= temp; return}      // move head

      while (p->next != 0) { q=p;    p=p->next;}
      q->next= temp;
      temp->next= p;
 }
```

3. Singly Linked List에서 가장 큰 숫자데이터를 삭제하는 "removeMax ADT"를 작성하시오

가정: List is not empty.
List before operation: 32 10 67 45 23
List after operation: 32 10 45 23
Operation returns 67

```
void List::removeMax() {
    Node* p, * max = head, * q = 0;
    p = head;

    while (p->next != NULL) {       // search through the last node
        if (max->data < p->next->data) {
            max = p->next;
            q = p;    // q= previous node
        }
        p = p->next;
    }
    if(q == 0)       head = max->next;      // if max is head node, move head
    else
        q->next = max->next;

    delete max;

}
```

4. [DLL] Homework4 (DLL exercise) 에서 "insertLast"와 "LocateCurrent" 를 작성하시오

1)
```
void insertLast(Type data)  {
    Node* temp = new Node(data);
    Node* p;

    if (head == 0)
        head = temp;
    else {
        p = head;
        while (p->next != 0)
            p = p->next;
        p->next = temp;
        temp->prev = p;
    }
    current = temp;
}
```

2) void List::locateCurrent(int Nth) {

```
Node* p;
int pos = 1;

if (head == 0)       cout << "List is empty!" ;

else if (listLength() >= Nth) {
      p = head;
      while (pos != Nth) {
            p = p->next;
            pos++;
      }
      current = p;
      cout << pos << " * ";
      cout << current->data;
}
else
      cout << "No such a line" << endl;
}
```

5. 다음 다항식을 아래 노드구조를 참조하여 일반리스트(Generalized List)로 표현하시오. (다항식: $9x^5 + 7x^4y + 10xz$)

Flag	coef	exp	next pointer

- Flag = 0 means *variable* is present
- Flag = 1 means *down pointer* is present
- Flag = 2 means *coefficient* and *exponent* is present

- head node는 3개의 field로 구성 (flag, variable, next pointer)

- temp1 can be read as x^4., temp2 = y temp3 = coefficient = 7
- temp1 x temp2: $x^4 \times 7y^1 = 7x^4y^1$