



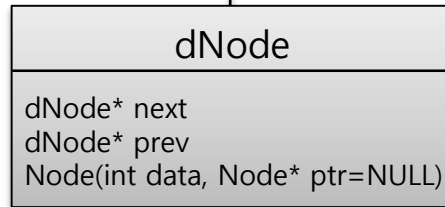
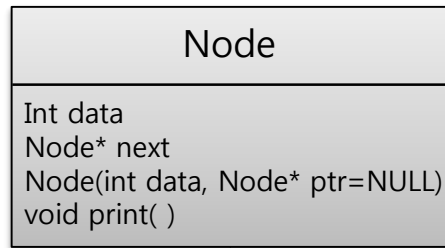
430.217

# Introduction to Data Structures

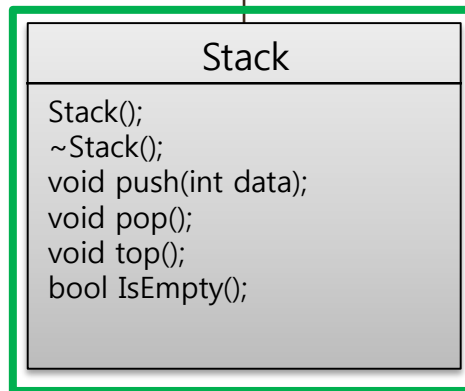
Assignment 2. dLinkedList / skip list

Seoul National University  
Advanced Computing Laboratory

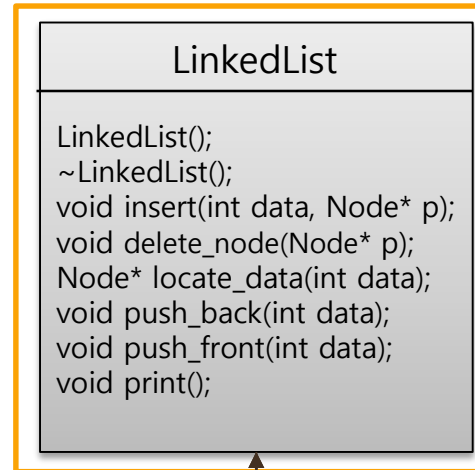
# Overview



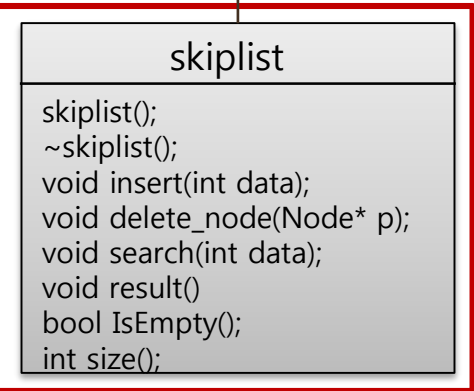
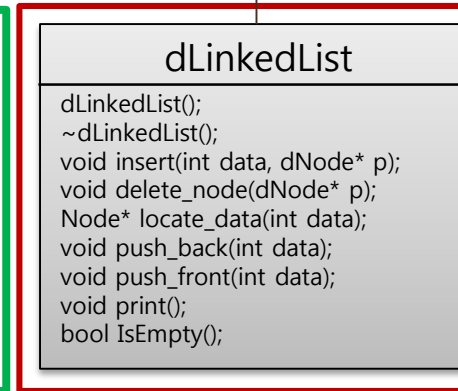
**Lab2**



**Lab1**



**hw2**



# 1. Doubly linked list

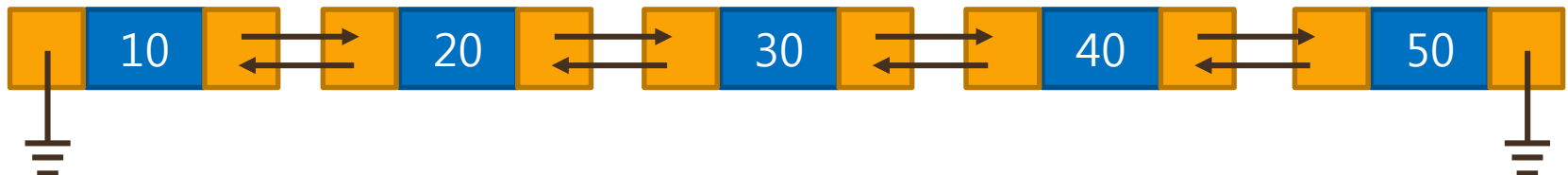
- Singly linked lists

- Only one direction
- Uses less memory per node



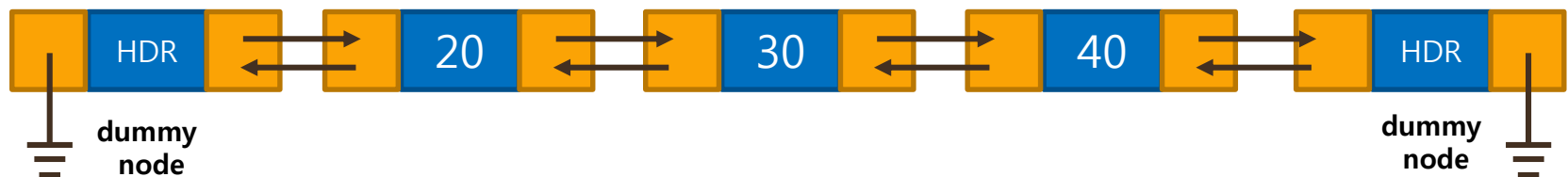
- Doubly linked list

- It can be iterated in reverse w/out recursion
- Faster performance
- Extra memory for pointer to previous node



# Implementation of doubly linked list

- Doubly linked list needs two dummy nodes (one at the head, one at the tail)
- dNode – derived from Node having two links next and prev
- Operations
  - push\_front(int data)
  - push\_back(int data)
  - pop\_front( )
  - pop\_back( )
  - Insert(int data, dNode\* p)
  - Delete\_node(dNode\* p)
  - locate\_data(int data)



# Implementation of dLinkedList

- Node를 상속받은 dNode class의 정의
- Operations
  - Insert(int data, dNode\* p)
  - delete\_node(dNode\* p)
  - locate\_data(int data)
  - push\_front(int data)
  - push\_back(int data)
  - pop\_front( )
  - pop\_back( )
  - print( )
  - isEmpty( )

\* header file 수정 시 주석 달 것

```
#include <iostream>
#include <string>
#include "Stack.h"

using namespace std;

class dNode:public Node
{
public:
    dNode* next;
    dNode* prev;
    dNode(int data, dNode* n_ptr = NULL, dNode* p_ptr = NULL);
};

class dLinkedList:public LinkedList
{
public:
    dLinkedList();
    ~dLinkedList();
    void push_front(int data);
    void push_back(int data);
    void pop_front();
    void pop_back();
    void insert(int data, dNode* p);
    void delete_node(dNode* p);
    dNode* locate_data(int data);
    void print();
    bool isEmpty();

private:
    dNode *Head, *Tail;
};
```

dLinkedList.h 예제

# Member functions

- void insert(int data, Node\* p)

```
void insert(int data, dNode* p);
```

- 구조체에서 특정 위치의 데이터의 값을 가지는 노드 삽입

- Void delete(Node\* p)

```
void delete_node(dNode* p);
```

- 구조체에서 특정 위치의 노드를 제거

- Node\* locate\_data(int data)

```
dNode* locate_data(int data);
```

- 구조체에서 데이터의 값을 가지는 노드의 위치를 출력

- void print( ) / bool IsEmpty( )

```
void print();
```

```
bool IsEmpty();
```

- Stack의 member function과 동일 (dNode의 특성상 제정의 필요)

# Member functions

- void push\_front(int data)

```
void push_front(int data);
```

- 구조체의 front(head)에 데이터의 값을 가지는 노드 삽입

- Void push\_back(int data)

- 구조체의 back(tail)에 데이터의 값을 가지는 노드 삽입

```
void push_back(int data);
```

- Void pop\_front( )

```
void pop_front();
```

- 구조체의 front(head)에서 노드 제거

- Void pop\_back( )

```
void pop_back();
```

- 구조체의 back(tail)에서 노드 제거

# Doubly linked list – practice main( )

```
int main()
{
    cout << "<Doubly linked list example>" << endl;
    dLinkedList* dlist = new dLinkedList();
    dlist->push_front(2);
    dlist->print();
    dlist->push_front(5);
    dlist->print();
    dlist->push_back(9);
    dlist->print();
    dlist->push_back(1);
    dlist->print();
    dNode* p = dlist->locate_data(2);
    dlist->insert(0, p);
    dlist->delete_node(p);
    dlist->print();
    return 0;
}
```

C:\Windows\system32\cmd.exe

<Doubly linked list example>

Doubly linked list:

2

Doubly linked list:

5

2

Doubly linked list:

5

2

9

Doubly linked list:

5

2

9

1

Doubly linked list:

5

0

9

1

계속하려면 아무 키나 누르십시오 . . .

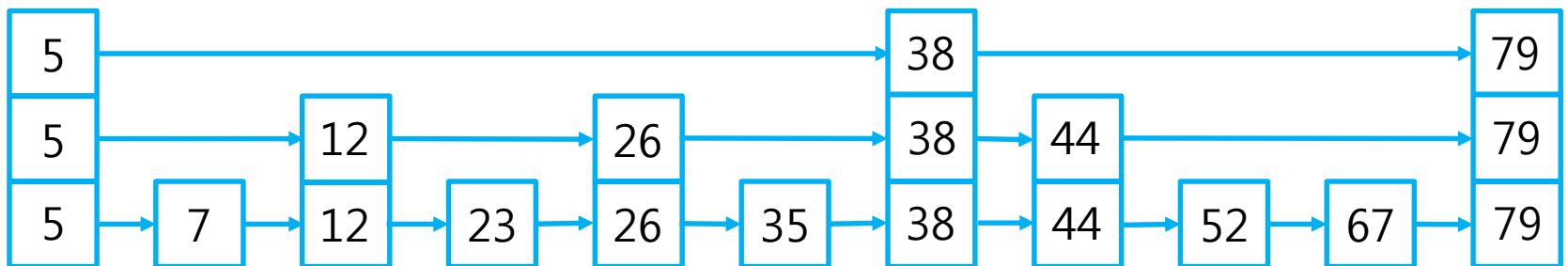


## 2. Skip list

- Linked list(랩1)를 사용하여 다음과 같은 데이터 구조를 생성한다.

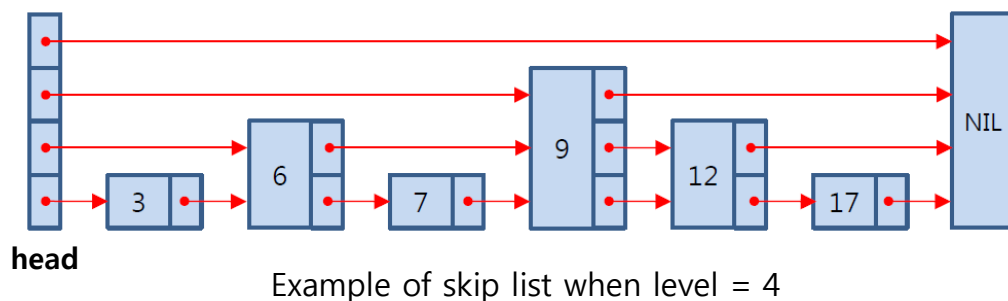


- Skip list 를 구현하고 다음과 같은 데이터 구조를 생성한다.



# Implementation of skip list

- 본 과제에서는 maximum node level = 3



- Operations
  - insert(int data)
  - delete\_node(int data)
  - search(int data)
  - int size( )

\* header file 수정 시 주석 달 것

```
#include <iostream>
#include <string>
#include "LinkedList.h"

using namespace std;

class skiplist : public LinkedList
{
public:
    skiplist();
    ~skiplist();
    void insert(int data);
    void delete_node(int data);
    Node* search(int data);
    void result();
    bool isEmpty();
    int size();
};
```

Skiplist.h 예제

# Member functions

- void insert(int data)

```
void insert(int data);
```

- data의 값을 skip list에 추가한다. 이미 skip list에 존재하면 error 메시지를 출력한다. (random으로 level의 수를 결정함)

- void delete\_node(int data);

```
void delete_node(int data);
```

- Skip list에 있는 data의 값을 지운다. 존재하지 않는 data이면 error 메시지를 출력한다.

# Member functions

- Node\* search (int data)

```
Node* search(int data);
```

- data의 값을 skip list에서 검색하여 주소 값을 출력한다. Data의 값이 skip list에 존재하지 않으면 error 메시지를 출력한다.

- void result()

```
void result();
```

- LinkedList 구조와 Skip List 구조에서의 insert, delete, search의 성능을 비교하여 출력한다.

- bool isEmpty( )

```
bool isEmpty();
```

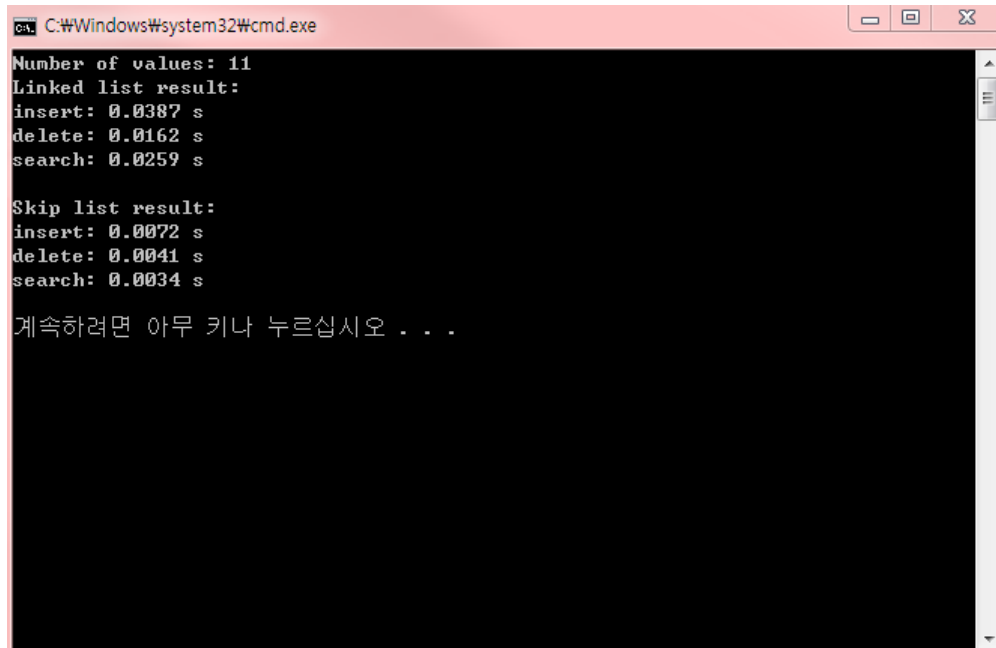
- int size();

```
int size();
```

- Skip list의 사이즈를 출력한다.

# Results

- 다음과 같은 operation을 수행하였을 때 Linked list와 Skip list 성능 비교
  - insert(50)
  - delete(35)
  - search(44)
- Output should look like:



```
C:\Windows\system32\cmd.exe
Number of values: 11
Linked list result:
insert: 0.0387 s
delete: 0.0162 s
search: 0.0259 s

Skip list result:
insert: 0.0072 s
delete: 0.0041 s
search: 0.0034 s

계속하려면 아무 키나 누르십시오 . . .
```

# 제출

- 문의 사항은 Q&A 이메일을 적극적으로 활용할 것 (snu.qna.ds@gmail.com)
- 코드 주석은 필수 - 부분점수
- 제출
  - ETL에서 assignment 2 에 제출할 것
  - Due: 10/18(일) 23:59
  - 위 assignment를 구현한 Visual studio 프로젝트 (내문서 안에 존재) 폴더 자체를 압축하여 제출
  - 압축파일 형식(학번.zip): 2015-12345.zip