시스템프로그래밍기초 실습 7주차

Structure

- Structure is a **user defined data type** available in C.

```
struct Book {
      char      title[50];
      char      author[50];
      char      subject[50];
      int       book_id;
};

struct Book book1;
```

```
typedef struct {
      char      title[50];
      char      author[50];
      char      subject[50];
      int       book_id;
} Book;

Book book2;
```

Structure

```
typedef struct {
    char    title[50];
    char    author[50];
    char    subject[50];
    int     book_id;
} Book;

Book book3;

book3.book_id

Book *ptr = &book3;

ptr->book_id
```

실습 예제 1) structure.c

structure.c

```c
#include <stdio.h>

/* Structure */
struct card {
    int pips;
    char suit;
};

/* Structure using typedef */
typedef struct {
    double re;
    double im;
} complex;

/* Nested structure */
struct dept {
    char name[25];
    int no;
};

typedef struct {
    char name[25];
    int employee_id;
    struct dept department;
    double salary;
} employee_data;

```

실습 예제 1) structure.c

```c
int main()
{
    struct card cards[4] = {{1, 'D'}, {2, 'S'}, {3, 'C'}, {4, 'H'}};

    employee_data a = {
                        "john",
                        3,
                        {"Engineering", 3},
                        1000
                      };

    printf("Name: %s\nid: %d\nDept: %s\nDept_no: %d\nSalary: %.2f\n",
            a.name,
            a.employee_id,
            a.department.name,
            a.department.no,
            a.salary);

    return 0;
}
```

실습 예제 1) structure.c 결과

```
spubuntu@sp:~/Downloads$ ./structure
Name: john
id: 3
Dept: Engineering
Dept_no: 3
Salary: 1000.00
```

malloc (stdlib.h)

- **memory allocation**

  사용할 메모리 공간을 확보.

  (주로, 동적할당시 사용되며, heap 영역에 메모리를 할당한다.)

```
void *malloc(size_t _Size);
```

  - parameter:
    - size – This is the size of the memory block, in bytes.
  - return value:
    - success: a pointer to the allocated memory
    - failure: NULL

free (stdlib.h)

- malloc 등으로 할당된 메모리 해제.

```
void free(void *ptr)
```

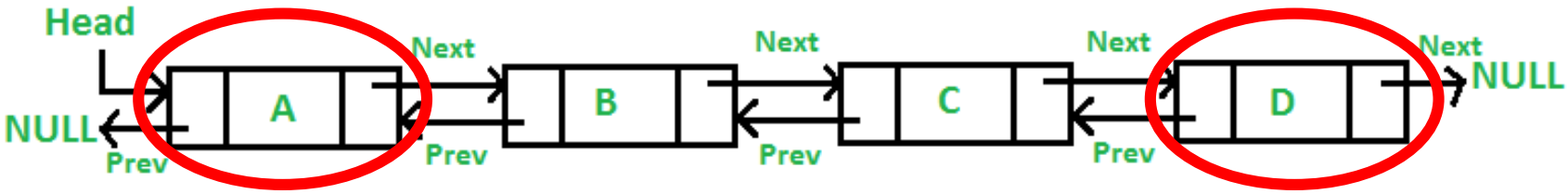
- parameter:
  - ptr – This is the pointer to a memory block previously allocated with malloc.

Node

Node

```
struct Node             // doubly linked list의 노드 구조체
{
  int data;             // 데이터를 저장할 멤버
  struct Node *next;    // 다음 노드의 주소를 저장할 포인터
  struct Node *prev;    // 이전 노드의 주소를 저장할 포인터
};
```

Doubly Linked List



HEAD Node:

list의 시작 노드

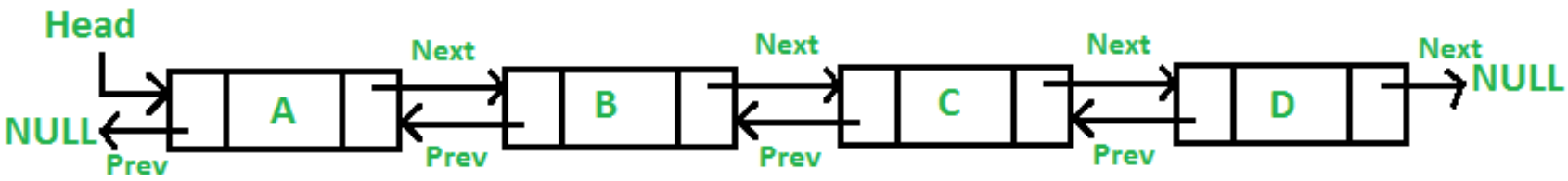
(Previous가 NULL)

TAIL Node:

list의 마지막 노드

(Next가 NULL)

Doubly Linked List



void insertAt(DLL *list, int index, Node *newnode)
: DLL(doubly linked list)의 특정 위치(index)에 newnode를 삽입.

void append(DLL *list, Node *newnode)
: DLL의 맨 뒤에 newnode를 삽입.

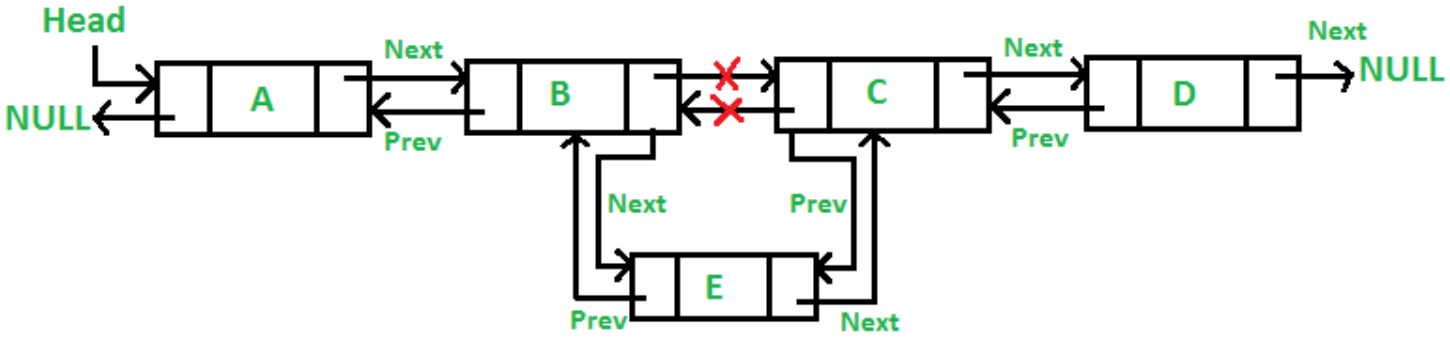
void deleteAt(DLL *list, int index)
: DLL 특정 위치(index)의 node를 삭제.

Doubly Linked List - insert

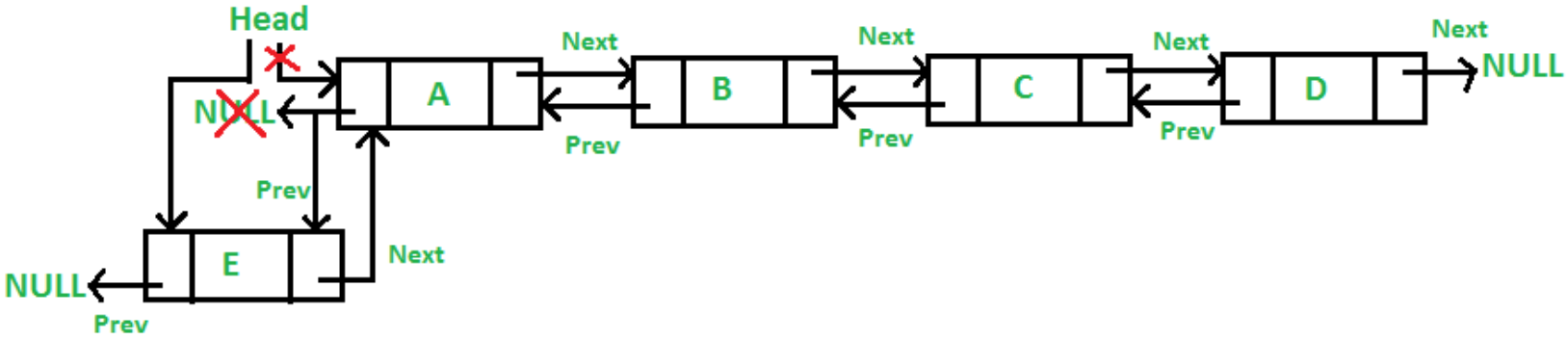
1) Add a node at the given index.

insertAt(DLL *list, int index, Node *newnode)

: DLL(doubly linked list)의 특정 위치(index)에 newnode를 삽입.



+) Add a node at the front

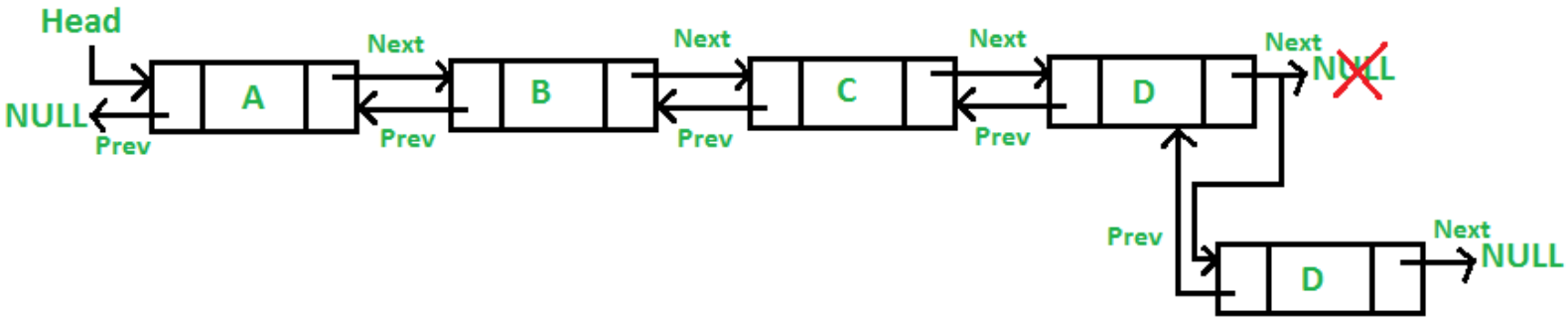


Doubly Linked List - insert

2) Add a node at the end

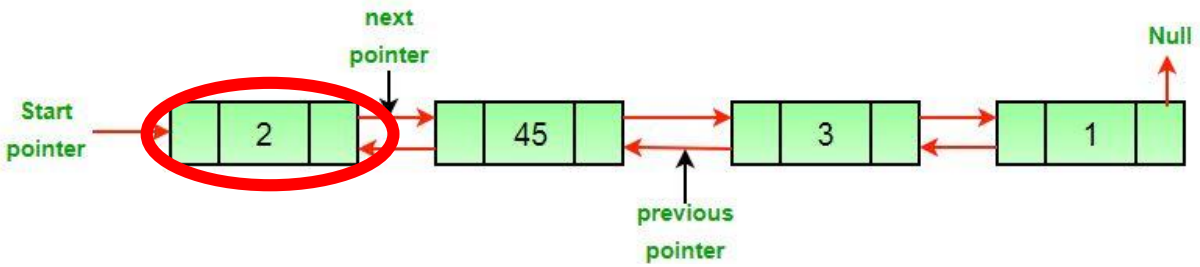
append(DLL *list, Node *newnode)

: DLL의 맨 뒤에 newnode를 삽입.

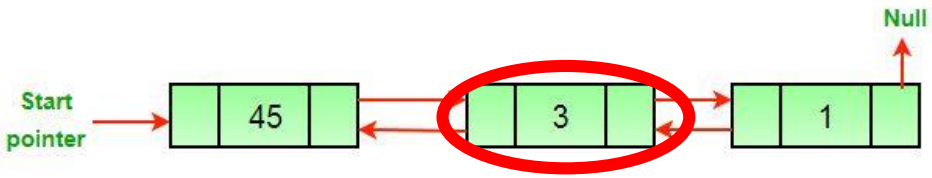


Doubly Linked List - delete

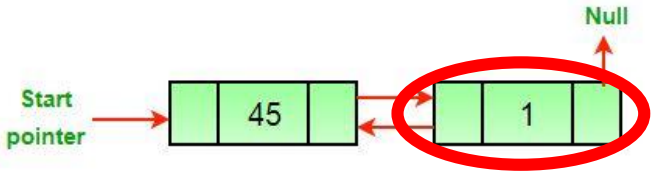
Original Doubly Linked List



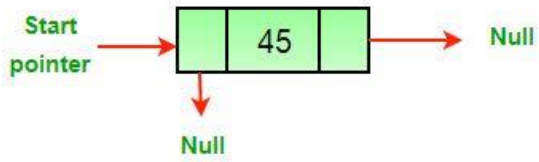
After deletion of head node



After deletion of middle node



After deletion of last node



실습 과제 1) dll.c

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Node{
    int val;
    struct Node *prev;
    struct Node *next;
} Node;

typedef struct {
    Node *head;
    int size;
} DLL;

Node *newnode(int n)
{
    Node *temp = (Node *)malloc(sizeof(Node));
    temp->val = n;
    temp->prev = NULL;
    temp->next = NULL;
    return temp;
}

DLL *newDLL()
{
    DLL *temp = (DLL *)malloc(sizeof(DLL));
    temp->head = NULL;
    temp->size = 0;
    return temp;
}
```

```c
/* TODO: implement following functions. */
void append(DLL *list, Node *newnode);
void insertAt(DLL *list, int index, Node *newnode);
void deleteAt(DLL *list, int index);
void print(DLL *list);
void print_reverse(DLL *list);
```

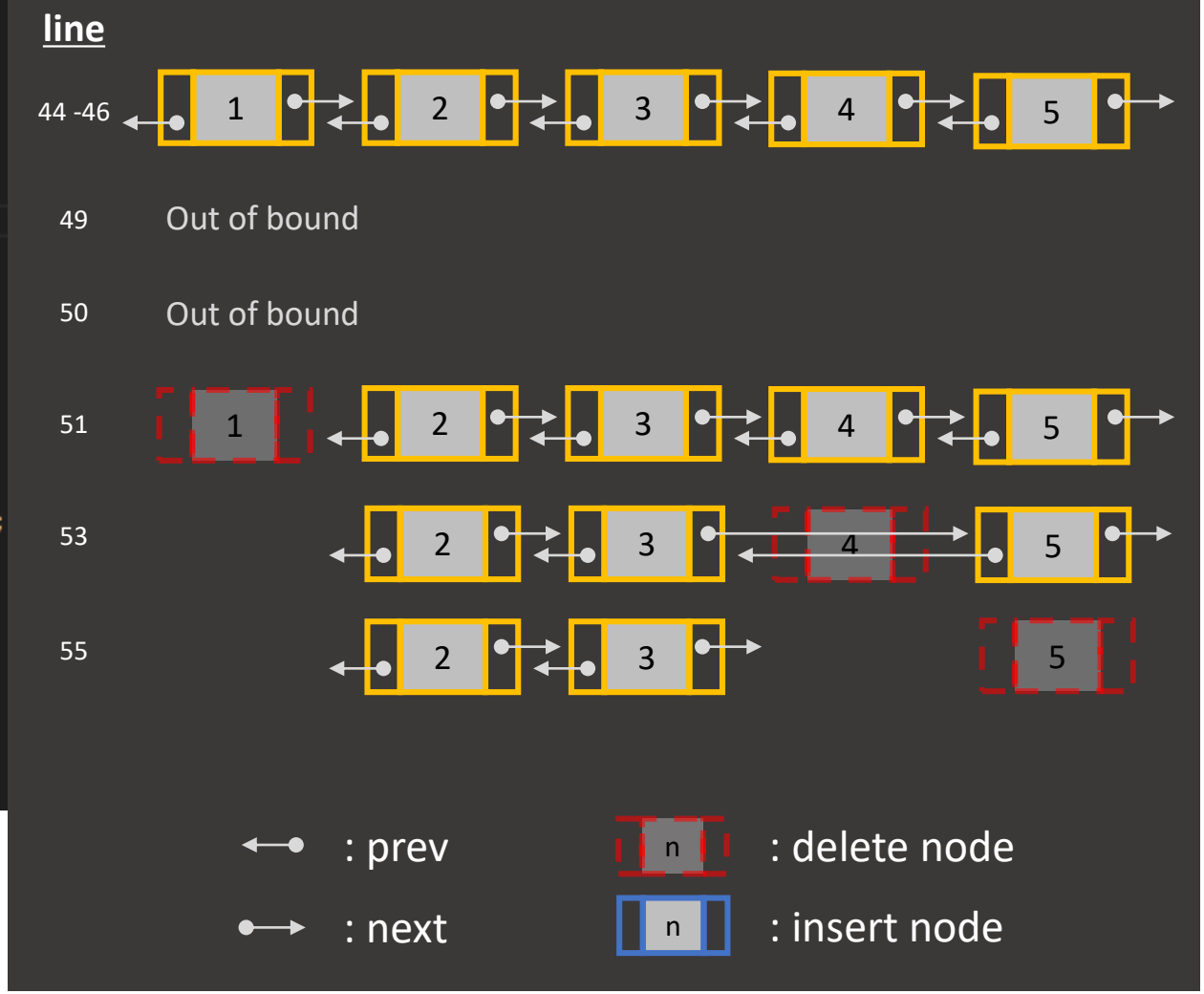
이 함수들을 구현할 것.

```c

int main()
{
    DLL *list = newDLL();
    int i;
    for (i = 1; i < 6; i++) {
        append(list, newnode(i));
    }
    print(list);

    deleteAt(list, -1);
    deleteAt(list, 5);
    deleteAt(list, 0);
    print(list);
    deleteAt(list, 2);
    print(list);
    deleteAt(list, 2);
    print(list);

    insertAt(list, -1, newnode(6));
    insertAt(list, 3, newnode(6));
    insertAt(list, 0, newnode(7));
    print(list);
    insertAt(list, 1, newnode(8));
    print(list);
    insertAt(list, 4, newnode(9));
    print(list);
    print_reverse(list);

    return 0;
}
```

실습 과제 1) dll.c

```
42      DLL *list = newDLL();
43      int i;
44      for (i = 1; i < 6; i++) {
45          append(list, newnode(i));
46      }
47      print(list);
48
49      deleteAt(list, -1);
50      deleteAt(list, 5);
51      deleteAt(list, 0);
52      print(list);
53      deleteAt(list, 2);
54      print(list);
55      deleteAt(list, 2);
56      print(list);
57
58      insertAt(list, -1, newnode(6));
59      insertAt(list, 3, newnode(6));
60      insertAt(list, 0, newnode(7));
61      print(list);
62      insertAt(list, 1, newnode(8));
63      print(list);
64      insertAt(list, 4, newnode(9));
65      print(list);
66      print_reverse(list);
```

Result of operation

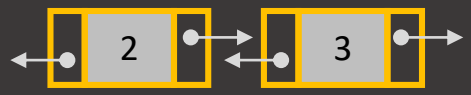


실습 과제 1) dll.c

```
42      DLL *list = newDLL();
43      int i;
44      for (i = 1; i < 6; i++) {
45          append(list, newnode(i));
46      }
47      print(list);
48
49      deleteAt(list, -1);
50      deleteAt(list, 5);
51      deleteAt(list, 0);
52      print(list);
53      deleteAt(list, 2);
54      print(list);
55      deleteAt(list, 2);
56      print(list);
57
58      insertAt(list, -1, newnode(6));
59      insertAt(list, 3, newnode(6));
60      insertAt(list, 0, newnode(7));
61      print(list);
62      insertAt(list, 1, newnode(8));
63      print(list);
64      insertAt(list, 4, newnode(9));
65      print(list);
66      print_reverse(list);
```

Result of operation

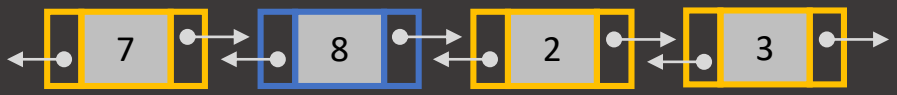
line

55 

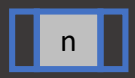
58 Out of bound

59 Out of bound

60 

62 

64 



실습 과제 1) dll.c 결과

```
spubuntu@sp:~/Downloads$ ./dll
[1] [2] [3] [4] [5]
DELETE ERROR: Out of Bound.
DELETE ERROR: Out of Bound.
[2] [3] [4] [5]
[2] [3] [5]
[2] [3]
INSERT ERROR: Out of Bound.
INSERT ERROR: Out of Bound.
[7] [2] [3]
[7] [8] [2] [3]
[7] [8] [2] [3] [9]
[9] [3] [2] [8] [7]
```

감사합니다.