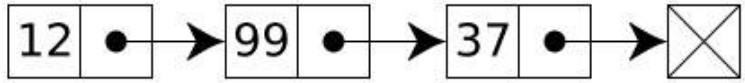
Programming Methodology
Practice Session #4

Linked List

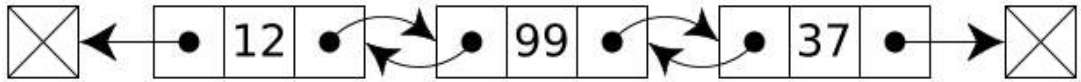
Linked List (1)

- Structure 데이터의 node들을 pointer로 선언된 **link를 통해 서로 연결한** list 형태의 자료 구조
- 한쪽 방향의 link만 있는 **singly-linked list**, 양 방향의 link가 모두 존재하는 **doubly-linked list**, list의 마지막 node의 link가 list의 처음을 가리키는 **circularly-linked list** 등의 다양한 type의 linked list가 있다.

Singly-linked List



Doubly-linked List



Circulary-linked List

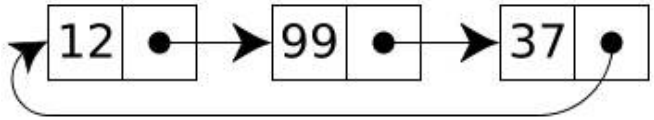
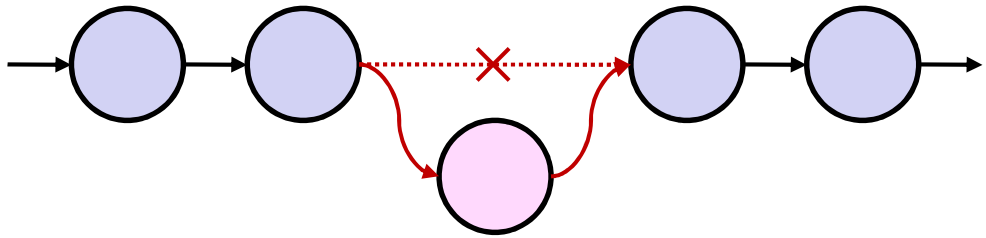


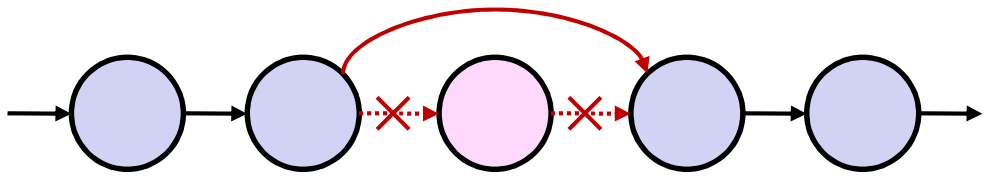
image 출처 : Wikipedia

Linked List (2)

- 기본적으로 array와 비슷한 자료 구조이나, 저장해야 할 데이터의 개수를 정확히 알 수 없을 경우에 linked list가 공간 절약에 유리하다.
- Linked list는 간단하게 node의 link를 변경함으로써 삽입과 삭제를 실행하므로, 삽입과 삭제가 빈번하게 일어나는 프로그램의 경우에 유리하다.
- Node의 삽입



- Node의 삭제



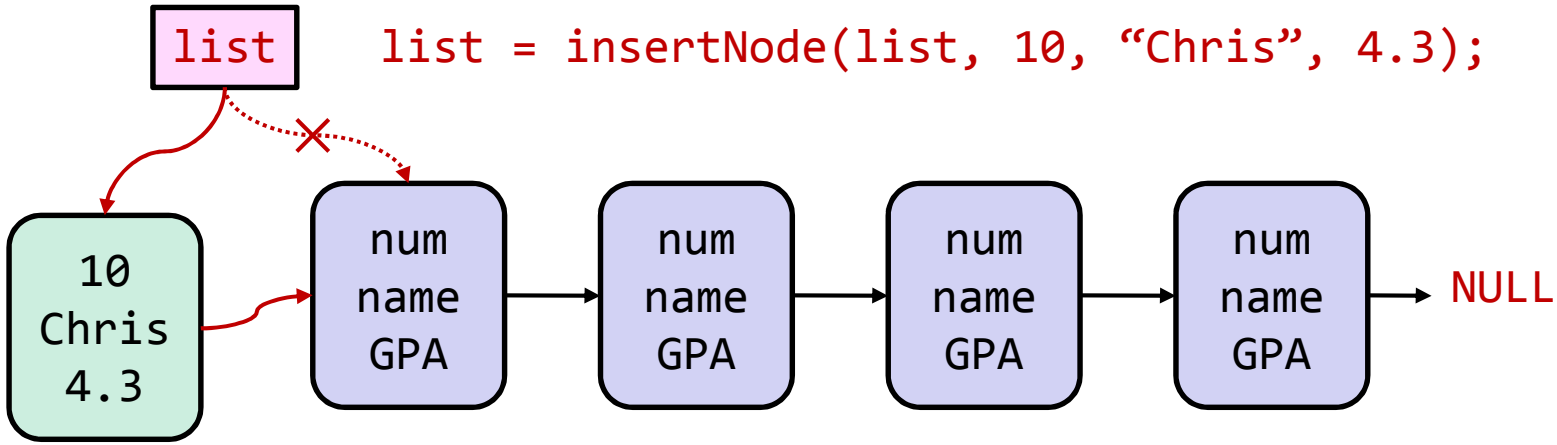
Sample Practice (1)

- **student** 구조체를 선언한다.
  - `int num;`
  - `char name[20];`
  - `double GPA;`
  - `struct student* next;`

- 다음과 같은 함수들을 구현한다.
  - `student* insertNode(student* list, int num, char* name, double GPA);`
  - `void printItem(student* item);`
  - `void printList(student* list);`
  - `void printTop(student* list);`
  - `student* mergeList(student* list1, student* list2);`

Sample Practice (2)

- `student* insertNode(student* list, int num, char* name, double GPA);`
  - `list`에 `num`, `name`, `GPA`의 데이터를 가지는 새로운 `student node`를 추가한다.
  - `list`의 head를 새로운 **node**로 교체하고, 이를 return한다.



Sample Practice (3)

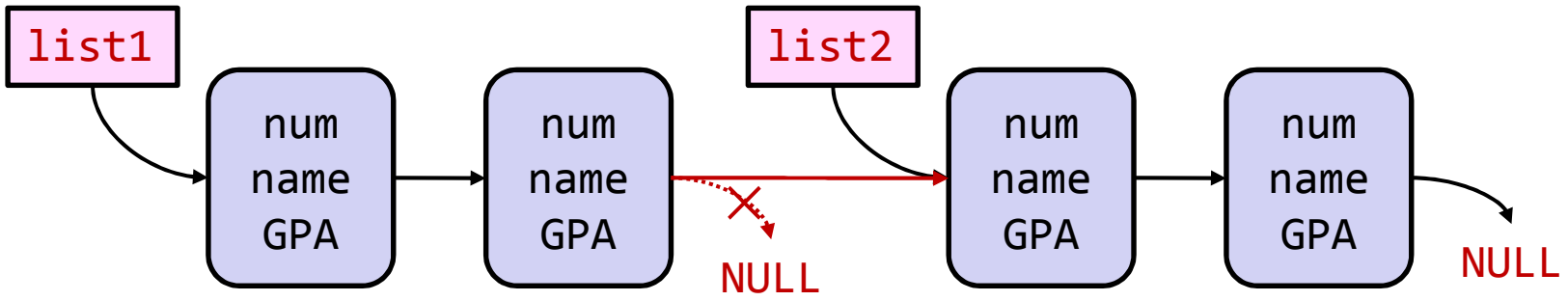
- `void printItem(student* item);`
  - 하나의 student node를 출력한다.
    - ex) [10] Chris : 4.2

- `void printList(student* list);`
  - list에 들어 있는 **모든 student node를 출력**한다.
  - printItem() 함수를 이용한다.
    - ex) [3] Elena : 4.0
      [2] George : 2.1
      [1] Albert : 3.2

Sample Practice (4)

- `void printTop(student* list);`
  - 최대값의 **GPA**를 가지는 student node를 찾아 이를 출력한다.
  - printItem() 함수를 이용한다.
    - ex) [3] Elena : 4.0

- `student* mergeList(student* list1,`
  `student* list2);`
  - **list1**의 뒤쪽에 **list2**를 붙이고, 합친 list의 head를 return한다.
  - 만약 list1이 NULL이라면, list2를 return한다.



Sample Practice (5)

```
void main()
{
    student* list1 = NULL;
    student* list2 = NULL;

    list1 = insertNode(list1, 1, "Albert", 3.2);
    list1 = insertNode(list1, 2, "George", 2.1);
    list1 = insertNode(list1, 3, "Elena", 4.0);

    list2 = insertNode(list2, 4, "Tommy", 2.6);
    list2 = insertNode(list2, 5, "Scarlet", 1.7);
    list2 = insertNode(list2, 6, "Chris", 4.2);

    printf("list1₩n");
    printList(list1);

    printf("₩nlist2₩n");
    printList(list2);

    printf("₩nTop of list1₩n");
    printTop(list1);

    printf("₩nTop of list2₩n");
    printTop(list2);

    printf("₩nMerge list1 and list2₩n");
    list1 = mergeList(list1, list2);
    printList(list1);
}
```

Sample Practice (6)

```
list1
[3] Elena : 4.0
[2] George : 2.1
[1] Albert : 3.2

list2
[6] Chris : 4.2
[5] Scarlet : 1.7
[4] Tommy : 2.6

Top of list1
[3] Elena : 4.0

Top of list2
[6] Chris : 4.2

Merge list1 and list2
[3] Elena : 4.0
[2] George : 2.1
[1] Albert : 3.2
[6] Chris : 4.2
[5] Scarlet : 1.7
[4] Tommy : 2.6
계속하려면 아무 키나 누르십시오 . . .
```