자료구조론

Linked List

u@hataeseong-ui-MacBook-Pro:~/Desktop/2017_CSE2010_2016025041/HW2$./hw2_2016025041 input.txt

Insertion(5) Failed : cannot find the location to be inserted
Deletion failed : element 9 is not in the list
Could not find 3 in the list
Key of the previous node of 7 is header.
Key of the previous node of 2 is 7.
key : 7 value : 3.140 key : 2 value : 3.140 key : 4 value : 3.140

u@hataeseong-ui-MacBook-Pro:~/Desktop/2017_CSE2010_2016025041/HW2$

- 실행 결과 : 예시와 일치

```
if(argc == 1)
        input = fopen("input.txt", "r");
else
        input = fopen(argv[1], "r");
header = MakeEmpty(header);
```

- 파일 오픈 및 헤더 생성

```
case 'i':
        fscanf(input, "%d %d", &key1, &key2);
        in.key = key1;
        tmp = Find(in, header);
        if(tmp != NULL) {
                printf("There already is an element with key %d. Insertion failed
                    \n", key1);
                break;
        }
         ElementType temp;
         temp.key = key2;
         tmp = Find(temp, header);
        Insert(in, header, tmp);
        break;
```

- 스위치 ~ 케이스문으로 입력 구문. 입력하려는 키가 이미 있는지 확인하고 없으면 넣을 키가 있는 위치를 찾아 삽입

```
void Insert(ElementType X, List L, Position P) {
    Position Tmp = NULL;
    if(X.key<0) {
        printf("Please use positive input. %d cannot be inserted\n", X.key);
        return;
    }
    X.value = 3.14;
    Tmp = (Position)malloc(sizeof(struct Node));
    Tmp->element = X;
    if(P==NULL) {
        printf("Insertion(%d) Failed : cannot find the location to be inserted\n",
        X.key);
        free(Tmp);
        return;
    }
    Tmp->next = P->next;
```

```
        P->next = Tmp;
    }
```

- 입력하려는 X의 value를 3.14로 설정 후 넣을 위치를 찾아 앞의 포인터를 Tmp로 원래 P의 포인터를 삽입할 노드의 포인트로 연결

```
    void PrintList(List L) {
        PtrToNode tmp = NULL;
        tmp = L->next;
        if(tmp==NULL) {
            printf("list is empty!\n");
            return;
        }
        while(tmp!=NULL) {
            printf("key : %d value : %.3f\t", tmp->element.key, tmp->element.value);
            tmp = tmp->next;
        }
        printf("\n");
    }
```

- 노드 포인터를 하나 만들어서 헤더부터 next포인터를 따라가면서 각 노드에 저장된 키와 값을 출력

```
    void Delete(ElementType X, List L) {
        Position P = NULL, Tmp = NULL;
        P = FindPrevious(X, L);
        if (!isLast(P, L)) {
            Tmp = Find(X, L);
            P->next = Tmp->next;
            free(Tmp);
        }
        else
            printf("Deletion failed : element %d is not in the list\n", X.key);
    }
```

- X가 저장되어 있는 노드를 Tmp에 저장한 후 앞뒤 포인터를 연결하고 Tmp 해제