Lab 3 & HW 3

Data Structure

Lab 3 (due on Lab Session)

1. Do p3_1.c

HW 3
(due on the day before the next Lab Session)

1. Do p3_2.c

Evaluation criteria

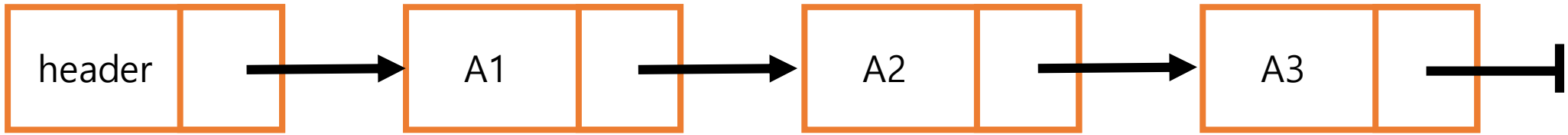
Category	Evaluation	
P3_1	50	
p3_2	50	
Total	100	

- *Use GCC 4.8 version or GCC 5.4 version.*
- *No score will be given if the gcc version is different.*

Lab3

- You should finish p3_1 (insert and print_list) during the lab session and submit it to git before you leave.
- For p3_2 (delete and find), you can submit it to the git later.
- Folder name : Lab3
- code name: p3_1, p3_2

Lab3 – Linked List



- **Insert** a new node right after the node with the given key. If your list does not have any node with the given key, just print an error message. (option i)
- **Delete** the node with the given key. If your list does not have any node wit(h the given key, just print an error message. (option d)
- **Find the previous node** of the node with the given key. If your list does not have any node with the given key, just print an error message. (option f)
- **Print the entire list**. If your list is empty, just print that your list is empty. (option p)

Lab3 – Linked List Inputs

- **i x y** insert a new node with the key “x” after the node with the key “y”
- **i x -1** insert a new node with the key “x” after the header node in the list.
- **d x** delete the node with the key “x”.
- **f x** print the key of the previous node of the node with the key “x”
- **p** print the entire list from the beginning to the end.

Lab3 – Linked List Implementation

- Structure

```
typedef struct Node *PtrToNode;
typedef PtrToNode List;
typedef PtrToNode Position;
typedef int ElementType;
struct Node
{
    ElementType   element;
    Position                  next;
};
```

- Function

```
<Lab3>
List MakeEmpty( List L );
int IsEmpty( List L );
int IsLast( Position P,  List L );
void Insert( ElementType X,  List L, Position P );
void PrintList(List L);

<HW3>
void Delete( ElementType X,  List L );
Position Find(ElemenType X, List);
Position FindPrevious ( ElementType X,  List L );
void DeleteList ( List L );
```

Lab3 – Linked List Implementation

- main(1)

```
int main(int argc, char *argv[]){
	char command;
	int key1, key2;
	FILE *input;
	Position header;
	Position tmp;
	if(argc == 1)
		input = fopen("input.txt", "r");
	else
		input = fopen(argv[1], "r");
	header = MakeEmpty(header);
	while(1) {
		command = fgetc(input);
		if(feof(input)) break;
		switch(command) {
		case 'i':
			fscanf(input, "%d %d", &key1, &key2);
			///tmp = Find(key2, header); hw3
			///Insert(key1, header, tmp); hw3
			Insert(key1,header,header);
			break;
```

Lab3 – Linked List Implementation

- main(2)

```
            case 'd':
                    fscanf(input, "%d", &key1);
                    Delete(key1, header);
                    break;
            case 'f':
                    fscanf(input, "%d", &key1);
                    tmp = FindPrevious(key1, header);
                    if(isLast(tmp, header))
                            printf("Could not find %d in the list₩n", key1);
                    else {
                            if(tmp->element>0)
                                    printf("Key of the previous node of %d is %d.₩n", key1, tmp->element);
                            else
                                    printf("Key of the previous node of %d is header.₩n", key1);
                    }
                    break;
            case 'p':
                    PrintList(header);
                    break;
            default:
                    break;
            }
    }
      DeleteList(header);
    fclose(input);
    return 0;
```

Lab3 – Linked List Result

- input file : lab2_input2.txt

```
i 3 -1
i 4 3
i 7 -1
i 5 8
d 3
i 2 7
d 9
f 3
f 7
f 2
p
```

- Result

```
yncho@ubuntu:~/tmp$ gcc p3_2.c -o p3_2
yncho@ubuntu:~/tmp$ ./p3_2 lab3_input2.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   key:2   key:4
```

Lab 3. List ADT – Insert, Show

- **Insert** a new node right after the node with the given key. If your list does not have any node with the given key, just print an error message
  - **i x -1 insert a new node with the key "x" after the head node in the list. (LAB)**
  - **i x y** insert a new node with the key "x" after the node with the key "y" (HW)
- **p** print the entire list from the beginning to the end.

```
<Function>
List MakeEmpty( List L );
int IsEmpty( List L );
int IsLast( Position P,  List L );
void Insert( ElementType X,  List L, Position P );
void PrintList(List L);
```

HW 3. List ADT – Insert, Show, Delete, Find

- **Delete** the node with the given key. If your list does not have any node with the given key, just print an error message.
- **Find the previous node** of the node with the given key. If your list does not have any node with the given key, just print an error message.
- **Show the entire list**. If your list is empty, just print that your list is empty.

```
<HW3>
void Delete( ElementType X,  List L );
Position Find( ElemenType X, List);
Position FindPrevious( ElementType X,  List L );
void DeleteList( List L );
```