LAB3

Lab3

- The deadline for lab3 submission is 22th March at 11:59 pm.

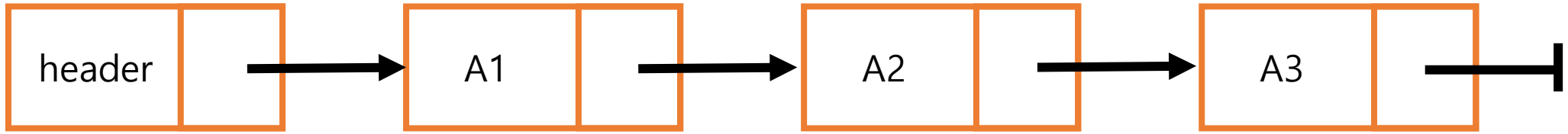
- Folder name : lab3
- code name: p3.c
- Each code will be tested by 5 different input files.
- 5 score for each input; if you don’t get the answer, you get 0 score.

Evaluation criteria

Category	Evaluation	
p3	100	
Total	100	

- *Use GCC **11.3** version.*
- *No score will be given if the gcc version is different.*

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. **Only positive number will be used as input.** (option i)
- **Delete** the node with the given key. If your list does not have any node with 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 – Inputs

- **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.
  - **i x y** insert a new node with the key “x” after the node with the key “y”.
- **p** print the entire list from the beginning to the end. If your list is empty, just print that “your list is empty.”
- **d x** delete the node with the key “x”. If your list does not have any node with the given key, just print an error message.
- **f x** print the key of the previous node of the node with the key “x”. If your list does not have any node with the given key, just print an error message.

Lab3 – Linked List Implementation

- Structure

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

typedef struct Node* PtrToNode;
typedef PtrToNode List;
typedef PtrToNode Position;
typedef int ElementType;

struct Node{
    ElementType element;
    Position next;
};
```

- Function

```c
List MakeEmptyList();
int isLast(Position p, List l);
void Delete(ElementType x, List l);
Position FindPrevious(ElementType x, List l);
Position Find(ElementType x, List l);
void Insert(ElementType x, Position p, List l);
void DeleteList(List l);
void PrintList(List l);
```

Lab3 – Example

- input file

```
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

```
root@0607fb0c13ae:/home/2022_ds/code/lab3# gcc -o p3 p3.c
root@0607fb0c13ae:/home/2022_ds/code/lab3# ./p3 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 previoud node of 7 is header.
Key of the previous node of 2 is 7.
key: 7  key: 2  key: 4
```

Lab3 – Sample code

```c
int main(int argc, char *argv[]) {

    char command;
    int key1, key2;
    FILE *input, *output;
    Position header = NULL, tmp = NULL;

    if(argc <= 1){
        printf("Please enter an input file.");
        return 0;
    }
    else
        input = fopen(argv[1], "r");


    header = MakeEmptyList();
    while(1){
        command = fgetc(input);
        if(feof(input)) break;

        switch(command){
        case 'i':
            fscanf(input, "%d %d", &key1, &key2);
            tmp = Find(key2, header);
            Insert(key1, tmp, header);
            break;
        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 previoud node of %d is header.\n", key1);
            break;
        case 'p':
            PrintList(header);
            break;
        defalut:
            printf("Invalid command line");
        }
    }

    DeleteList(header);
    fclose(input);

    return 0;
}
```

Lab3 – Sample code

```
List MakeEmptyList(){
    List l = (List)malloc(sizeof(struct Node));
    l->element = -999;
    l->next = NULL;
    return l;
}

int isLast(Position p, List l){
    Position current = l;
    while(current->next != NULL)
        current = current->next;
    return p == current;
}

//returns key of the previous node of x in list l.
Position FindPrevious(ElementType x, List l){

}

//returns the position of the key x in l
Position Find(ElementType x, List l){

}

//insert x after p in l
void Insert(ElementType x, Position p, List l){

}

//delete x in l
void Delete(ElementType x, List l){

}
```

```
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\t", tmp->element);
        tmp = tmp->next;
    }
    printf("\n");
}

void DeleteList(List l){
    Position p = NULL, tmp = NULL;
    p = l->next;
    l->next = NULL;
    while(p != NULL)
    {
        tmp = p->next;
        free(p);
        p = tmp;
    }
}
```