Introduction to Data Structures

Jinkyu Lee

Dept. of Computer Science and Engineering,
Sungkyunkwan University (SKKU)

Homework 3A

- 40 points for coding evaluation
  - Submission format
    - File name: yourid_HW3A.c
      - Example: 2000123456_HW3A.c
    - File type: .c (NOT .cpp)
  - Submission site: https://icampus.skku.edu
    - Week 9: [Homework] 3A (code)

- 1 point for report
  - The report is not evaluated in detail but evaluated as Pass/Fail
  - Template: Homework Report Template.docx
  - Submission format
    - File name: yourid_HW3A.pdf
      - Example: 2000123456_HW3A.pdf
    - File type: .pdf
  - Submission site: https://icampus.skku.edu
    - Week 9: [Homework] 3A (report)

- Due date
  - 11/10 23:59 (no late submission accepted)

Rules for homework

- You should follow instructions.
  - Complier
    - You will get no/less point if your program cannot be complied with the specified complier
  - Input/output format
    - You will get no/less point if TA's automatic evaluation program cannot parse your input or output.
  - Permitted modification scope
    - You will get no/less point if you modify code outside of the permitted modification scope
  - All other rules
    - You will get severe penalty or no/less point if you violate the given rules.

Complier and input/output rules for homework

- Every implementation homework will be evaluated by TA's automatic evaluation program with the following complier.
  - Complier: GCC 7.X, 8.X, 9.X or 10.X
    - https://gcc.gnu.org/
  - You will get no/less point if your program cannot be complied with GCC 7.X, 8.X, 9.X or 10.X.
    - For example, do not rely on visual studio.
  - You can use standard library such as *stdlib.h* and *math.h*.
- Input/output format
  - You will get no/less point if TA's automatic evaluation program cannot parse your input or output according to the following rules.
  - Use `stdin` and `stdout`

Problem

- Problem: Implementation of sorted linked list for octal number
  - Octal numbers are 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, …
  - The linked list is sorted in an ascending order.
  - Make the basic functions and the merge function for the sorted linked list for octal number.
  - Each octal number is stored as an integer value, and the range is 0 ~ 777.
  - Print the sorted linked list as an octal number AND a hexadecimal number.
    - Hexadecimal numbers are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, …
  - The length of each line of input is at most 50 integer values separated by a blank.

Input/Output

- Input

5 32 10 77 132 5 ----------> List 1
3 13 321 55 ----------> List 2

The number of nodes in each list

- Output

5 10 32 77 132 ----------> List 1(sorted)
13 55 321 ----------> List 2(sorted)
5 10 13 32 55 77 132 321 ----------> Merged list(sorted) as octal
5 8 B 1A 2D 3F 5A D1 ----------> Merged list(sorted) as hexa

Not 05, 08, 0B

Template

You should property insert a node so as to maintain a sorted list

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/* Modify from here */


/* Modify to here */


typedef enum {false, true} bool;


typedef struct _Node
{
      int octal;
      struct _Node* next;
} Node;

typedef struct
{
      Node* head;
      int len;
} LinkedList;

void InitList(LinkedList* plist);
int IsEmpty(LinkedList* plist);
void Insert(LinkedList* plist, int data);
void MergeTwoList(LinkedList* plist_a, LinkedList* plist_b,
LinkedList* plist_c);
void PrintList(LinkedList* plist);
void PrintListHex (LinkedList* plist);
/* Modify from here */


/* Modify to here */
```

```
int main() {
    …

    InitList(&list_a);
    InitList(&list_b);
    InitList(&list_c);

    for(i=0; i<2; i++)
    {
        scanf("%d",&cnt);
        for(j=0; j<cnt; j++)
        {
            scanf("%d",&input);
            if(i == 0)
                Insert(&list_a,input);
            else
                Insert(&list_b,input);
        }
    }
    MergeTwoList(&list_a,&list_b, &list_c);

    PrintList(&list_a);
    PrintList(&list_b);
    PrintList(&list_c);
    PrintListHex(&list_c);

    return 0;
}


/* Modify from here */


/* Modify to here */
```

Template

- You cannot modify the template except the space between /*Modify from here*/ and /*Modify to here*/
  - Do not remove /*Modify from here*/ and /*Modify to here*/
  - TA will copy the space and evaluate your code.
  - You may add user-defined functions and header files between /*Modify from here*/ and /*Modify to here*/.
  - In the space, you need to implement the following functions. (Next page)

Template

- void InitList(LinkedList* plist)
  - Initialize the list
- int IsEmpty(LinkedList* plist);
  - Check weather the list is empty or not
- void Insert(LinkedList* plist, int data);
  - Insert a node to the list; after inserting, the list should be sorted in an ascending order.
- void MergeTwoList(LinkedList* plist_a, LinkedList* plist_b, LinkedList* plist_c);
  - Merge two list; after merging, the resulting list should be sorted in an ascending order.
- void PrintList(LinkedList* plist);
  - Print the octal number in each node sequentially
- void PrintListHex (LinkedList* plist);
  - Print the hexadecimal value of the octal number in each node sequentially

Evaluation

- Evaluation
  - TA will test several cases.
  - Read Pages 7~9 (regarding template) carefully.
  - For each test case,
    - If your C code results in an answer within 10 seconds on a platform with average computing power,
      - If your output is perfect for four lines,
        - You get 100%.
      - Else if your output is valid for only for X lines (1<=X<=3)
        - You get 25%*X.
      - Else,
        - You get 0%.
    - Else,
      - You get 0%.