C Programming

Practice 11

Dynamic memory allocation - malloc()

Syntax for malloc()
ptr = (cast type *)malloc(byte size);

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
    int *pi;
    pi = (int*)malloc(sizeof(int));
    *pi = 3;
    printf("%d₩n", *pi);
    free(pi);
    return 0;
}
```

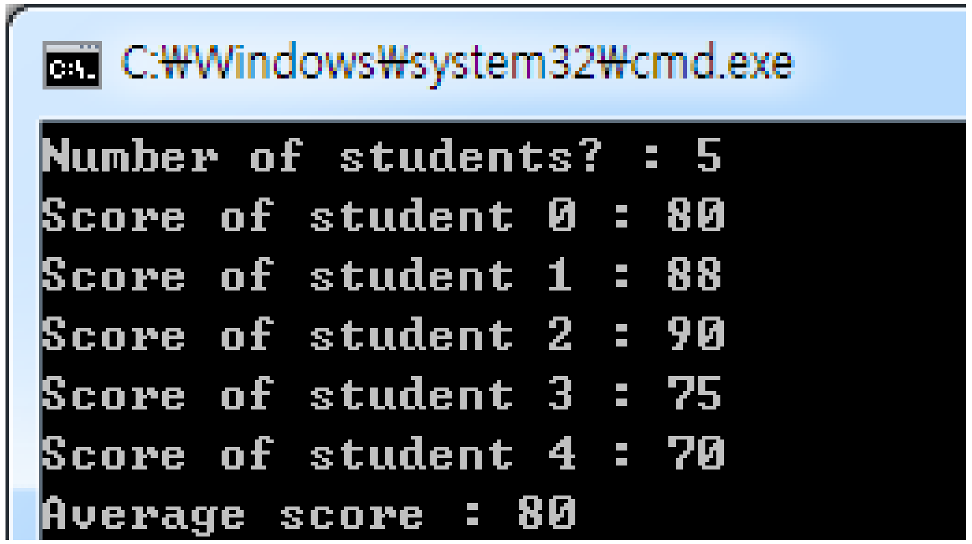
Dynamic memory allocation - free()

Syntax for free()
free(ptr);

```c
#include <stdio.h>
#include <stdlib.h>
 int main()
{
    int *pi;
    pi = (int*)malloc(sizeof(int));
    *pi = 3;
    printf("%d₩n", *pi);
    free(pi);
    return 0;
}
```

Homework 22 – Dynamic allocation

- Receives a number of students and scores
- Print the average score of students
- Use malloc() and free() functions

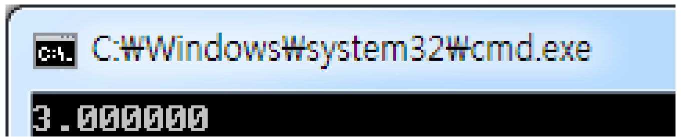


C:\Windows\system32\cmd.exe

```
Number of students? : 5
Score of student 0 : 80
Score of student 1 : 88
Score of student 2 : 90
Score of student 3 : 75
Score of student 4 : 70
Average score : 80
```

typedef

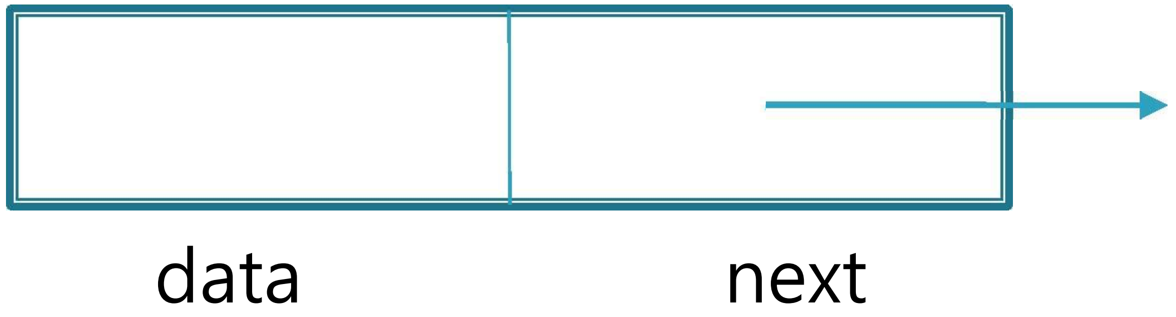
```
#include <stdio.h>
#include <stdlib.h>

#define N 3
typedef double scalar;
typedef scalar vector[N];
typedef vector matrix[N];

int main(int argc, char **argv)
{
    scalar a; // double a;
    vector b; // double b[3];
    matrix c; // double c[3][3];

    a = 1;

    b[0] = 1; b[1] = 2; b[2] = 3;

    c[1][1] = 3;

    printf("%f\n", c[1][1]);

    return 0;
}
```


C:₩Windows₩system32₩cmd.exe

```
3.000000
```

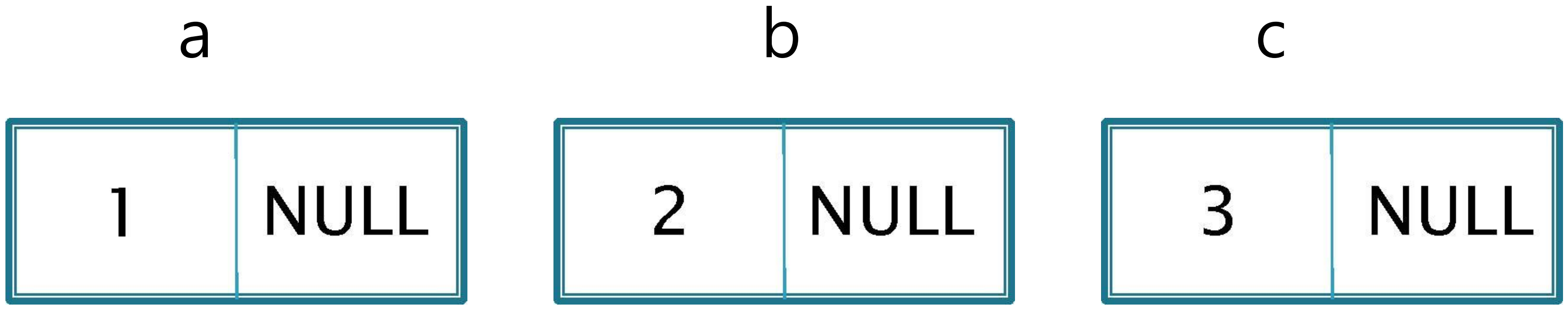
Linked-list

```
struct list {
        int            data;
        struct list *next;  /* called a link */
    };
```



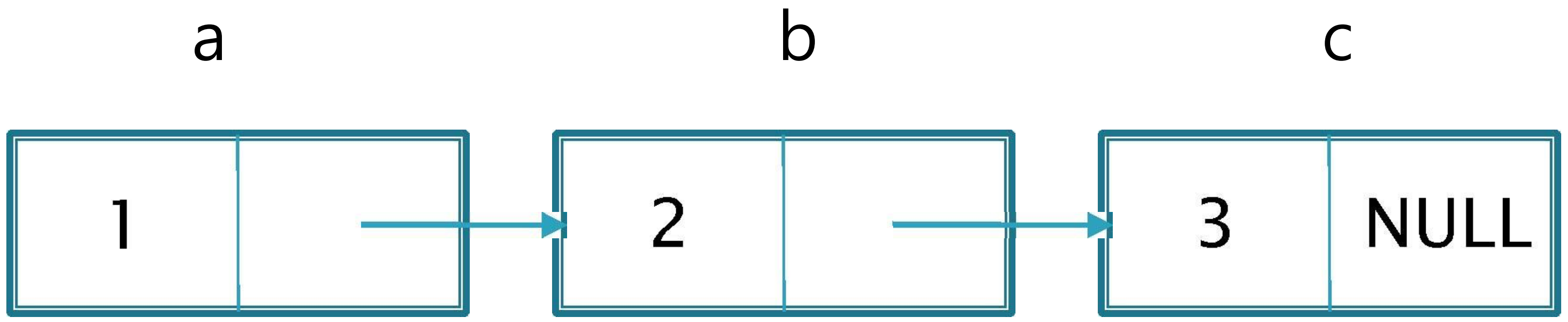
Linked-list

```
struct list  a, b, c;
a.data= 1;
b.data= 2;
c.data= 3;
a.next= b.next= c.next= NULL;
```



Linked-list

```
a.next=  &b;
b.next=  &c;
```

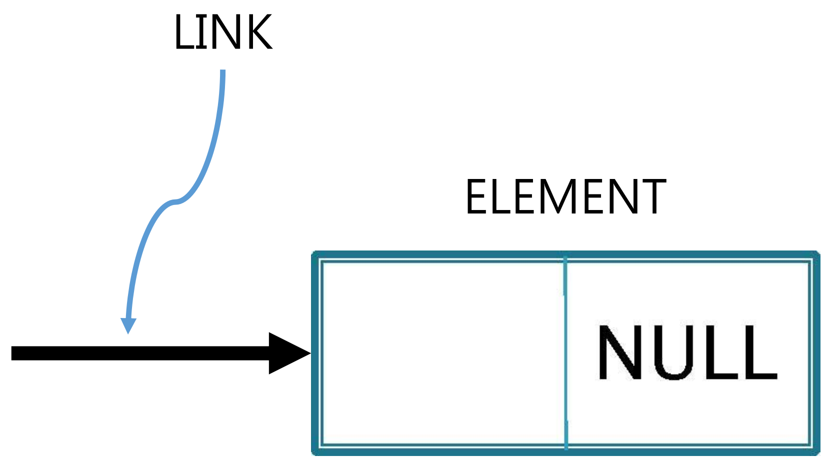


Basic List Operations

- Creating a list
  ```
  LINK string_to_list(char  s[]);
  ```
- Counting the elements
  ```
  int count(LINK  head);
  ```
- Looking up an element
  ```
  LINK lookup(DATA  c,  LINK  head);
  ```
- Inserting an element
  ```
  void insert(LINK  p1,  LINK  p2,  LINK  q);
  ```
- Deleting an element
  ```
  void delete_list(LINK  head);
  ```

Linked-list structure

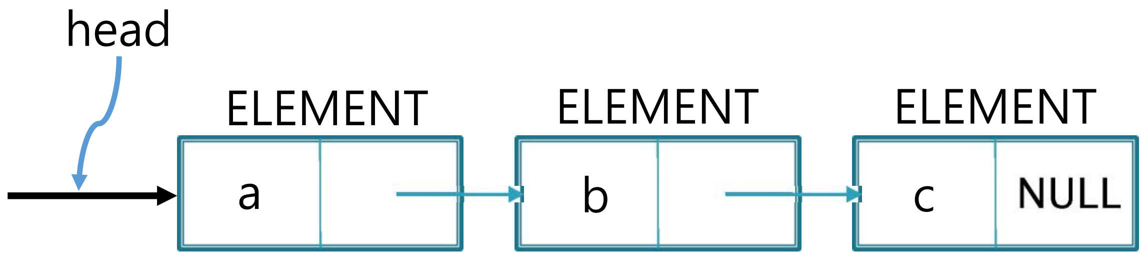
```
#include <stdio.h>  /* NULL is defined here */
#include <stdlib.h>
typedef char  DATA; /* use char in examples */
struct linked_list{
    DATA                d;
    struct linked_list *next;
};
typedef struct linked_list  ELEMENT;
typedef ELEMENT *           LINK;
```



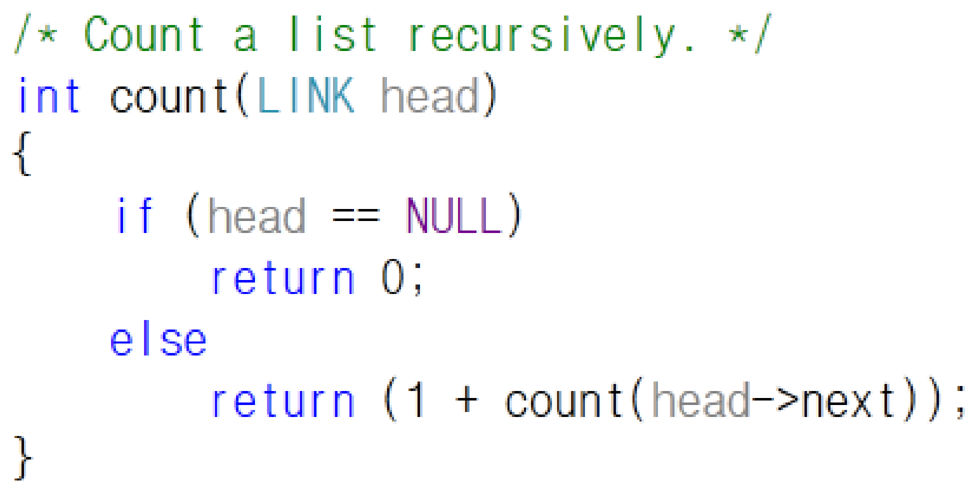
Creating a List

```
LINK string_to_list(char s[])
{
    LINK head;
    if(s[0] == '\0')
        return NULL;
    else
    {
        head = (LINK)malloc(sizeof(ELEMENT));
        head->d = s[0];
        head->next = string_to_list(s + 1);
        return head;
    }
}
```

s[4] = {'a', 'b', 'c', '₩0'}

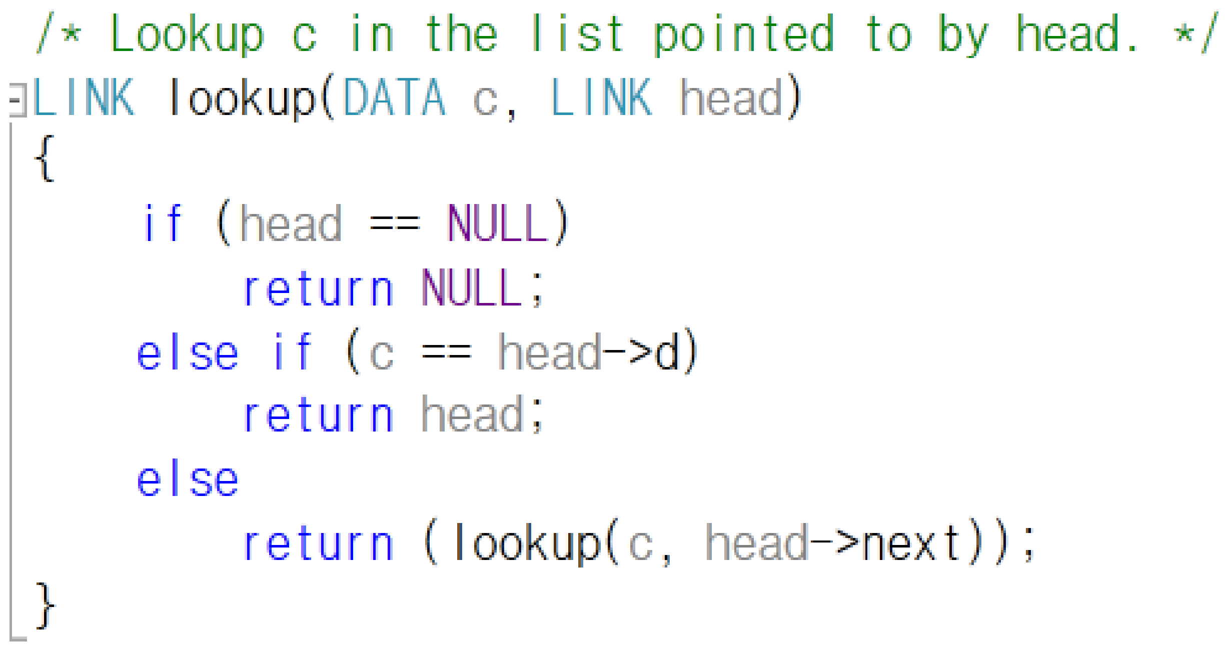


Counting the elements



```c
/* Count a list recursively. */
int count(LINK head)
{
    if (head == NULL)
        return 0;
    else
        return (1 + count(head->next));
}
```

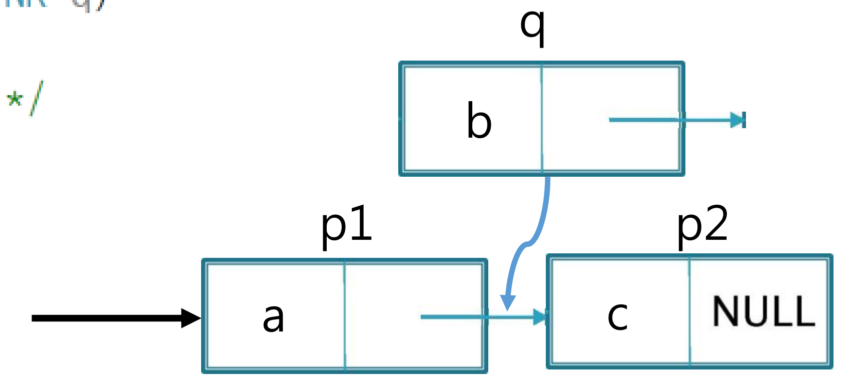
Looking up an element



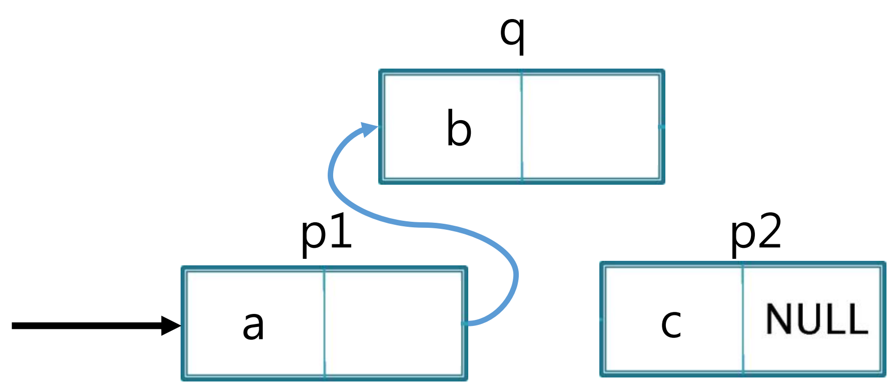
```
/* Lookup c in the list pointed to by head. */
LINK lookup(DATA c, LINK head)
{
    if (head == NULL)
        return NULL;
    else if (c == head->d)
        return head;
    else
        return (lookup(c, head->next));
}
```

Inserting an element

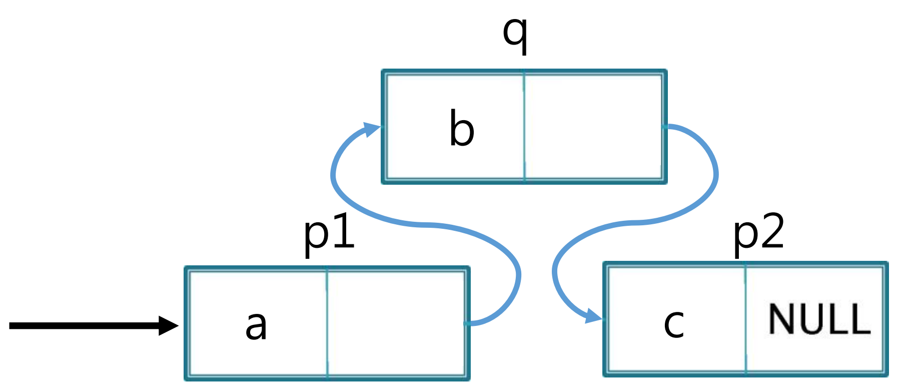
```
/* Insert an element in a linked list:
by having two adjacent elements pointed at by
p1 and p2 and by inserting between them an element pointed at by q.*/
void insert(LINK p1, LINK p2, LINK q)
{
  ① p1->next = q;   /* insertion */
  ② q->next = p2;
}
```



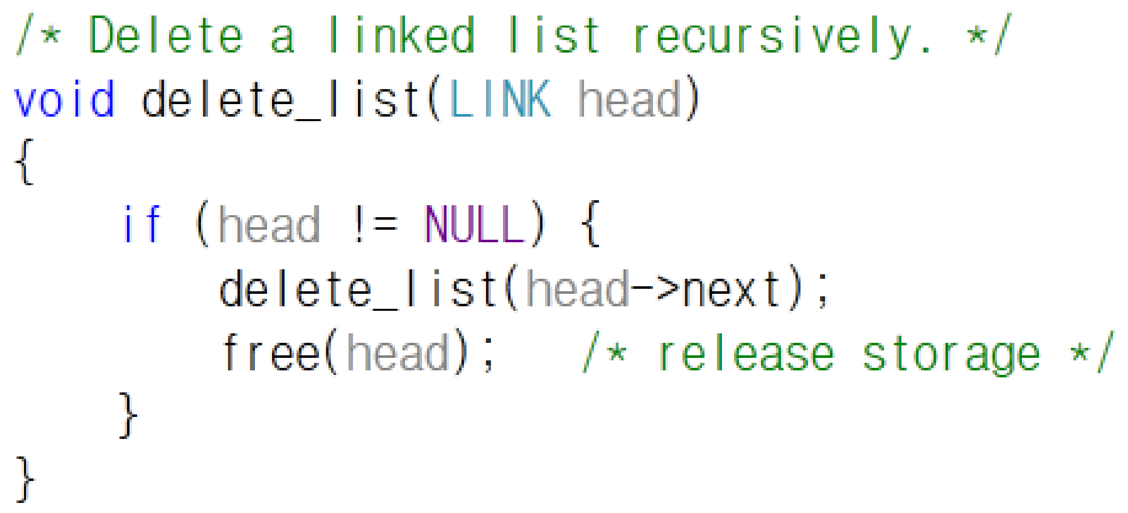
①



②



Deleting an element



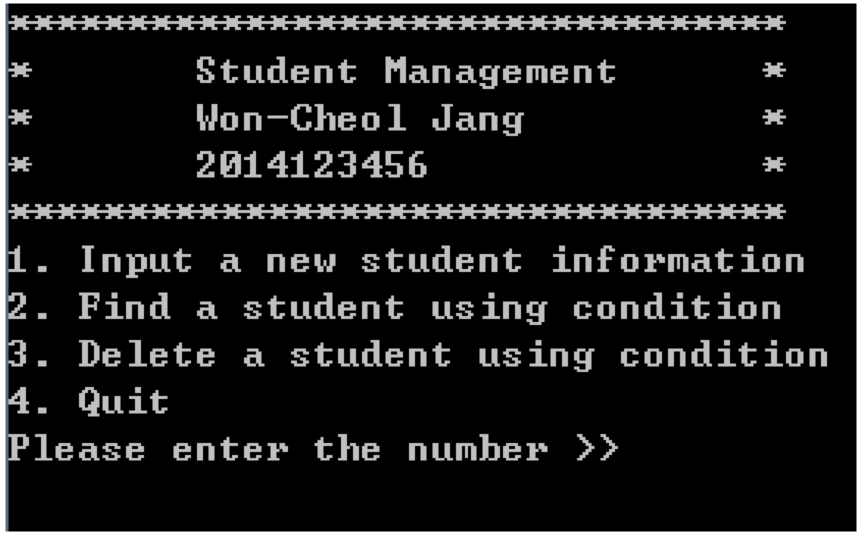
```
/* Delete a linked list recursively. */
void delete_list(LINK head)
{
    if (head != NULL) {
        delete_list(head->next);
        free(head);    /* release storage */
    }
}
```

Project 3 - Student management program (12/10, 5 points)

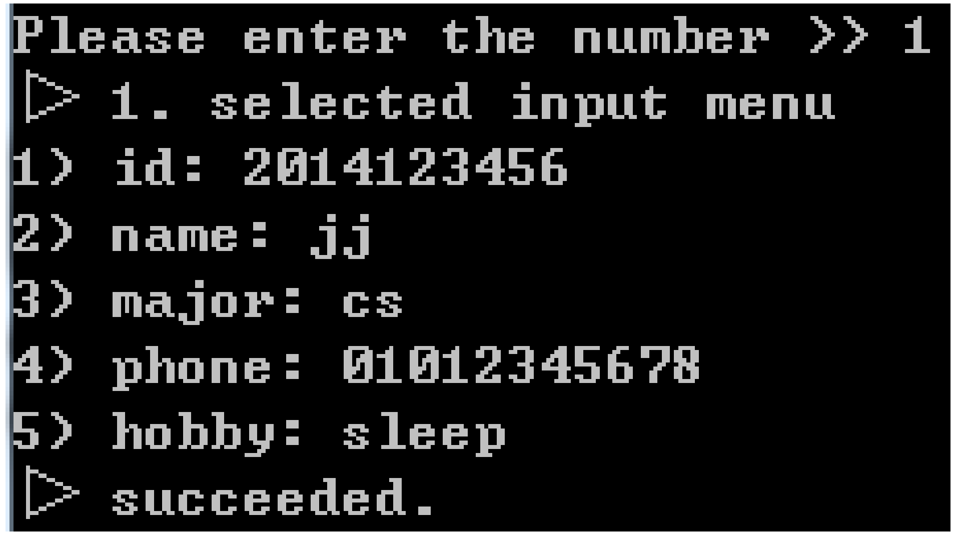
```
typedef struct record
{
        char id[20];
        char name[20];
        char major[20];
        char phone[20];
        char hobby[20];
        struct record * next;
} STUDENT;
```

Project 3 - Student management program (12/10, 5 points)

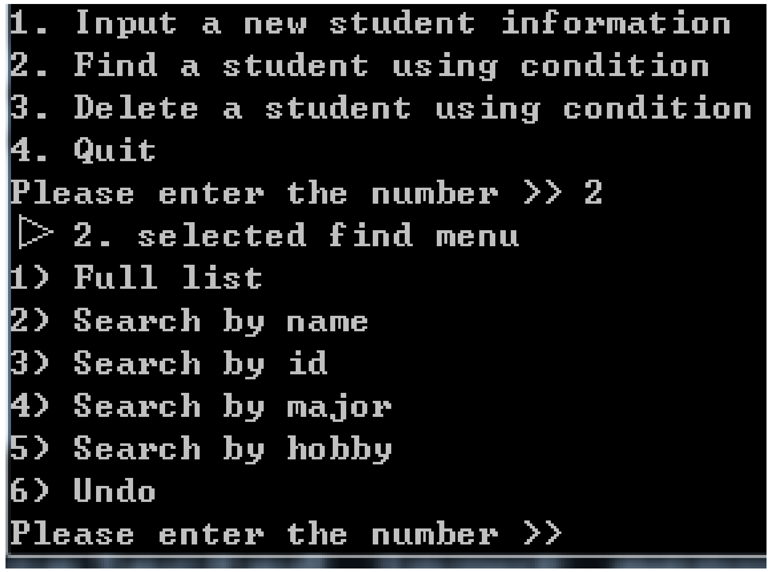
- Use a given structure
- Student management program must have functions (Input, Find, Delete, Quit)



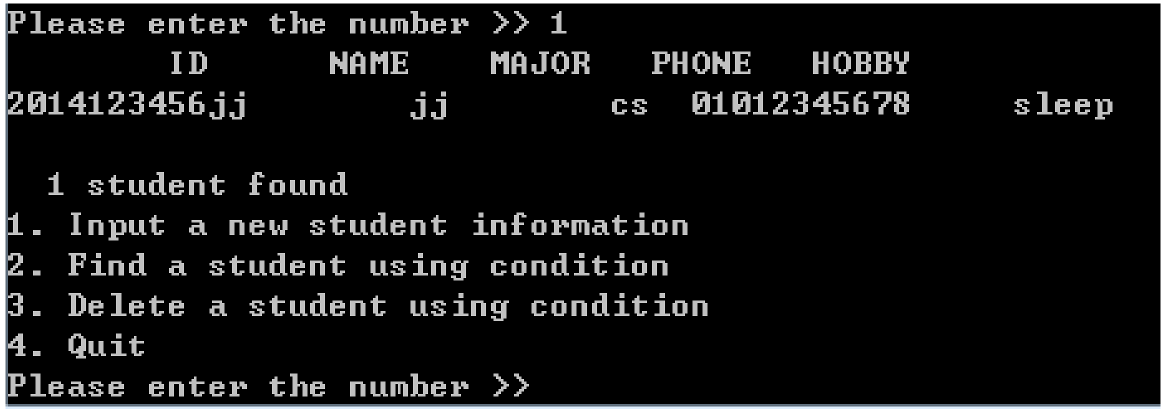
```
*********************************
*       Student Management      *
*       Won-Cheol Jang          *
*       2014123456              *
*********************************
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >>
```



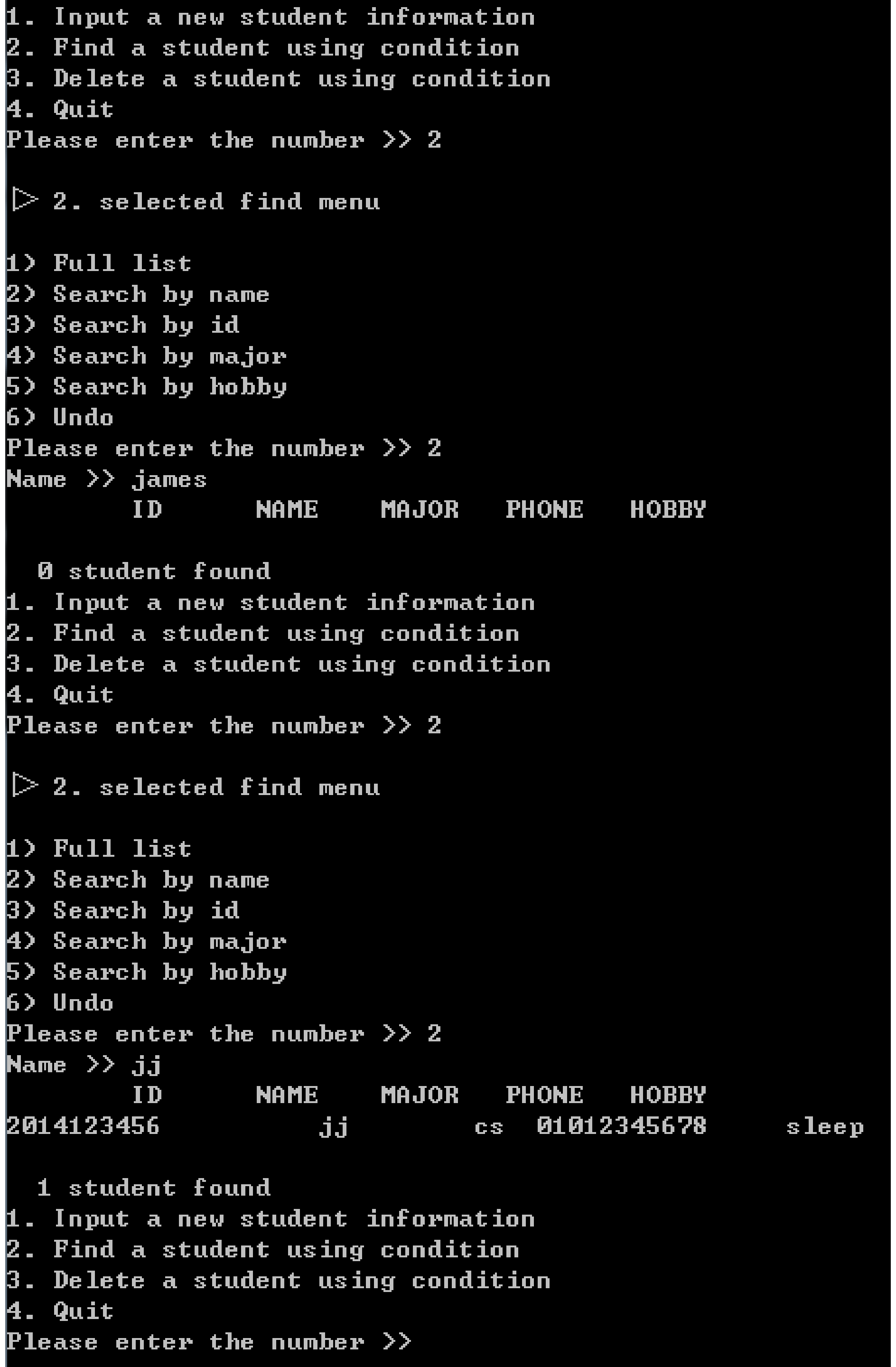
```
Please enter the number >> 1
▷ 1. selected input menu
1) id: 2014123456
2) name: jj
3) major: cs
4) phone: 01012345678
5) hobby: sleep
▷ succeeded.
```



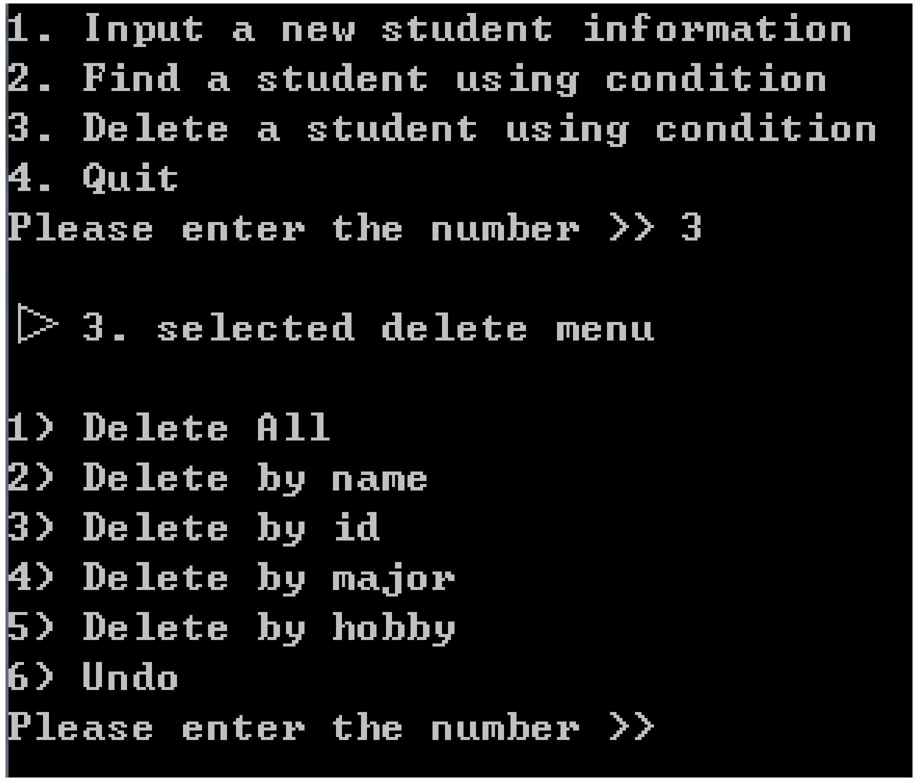
```
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >> 2
▷ 2. selected find menu
1) Full list
2) Search by name
3) Search by id
4) Search by major
5) Search by hobby
6) Undo
Please enter the number >>
```



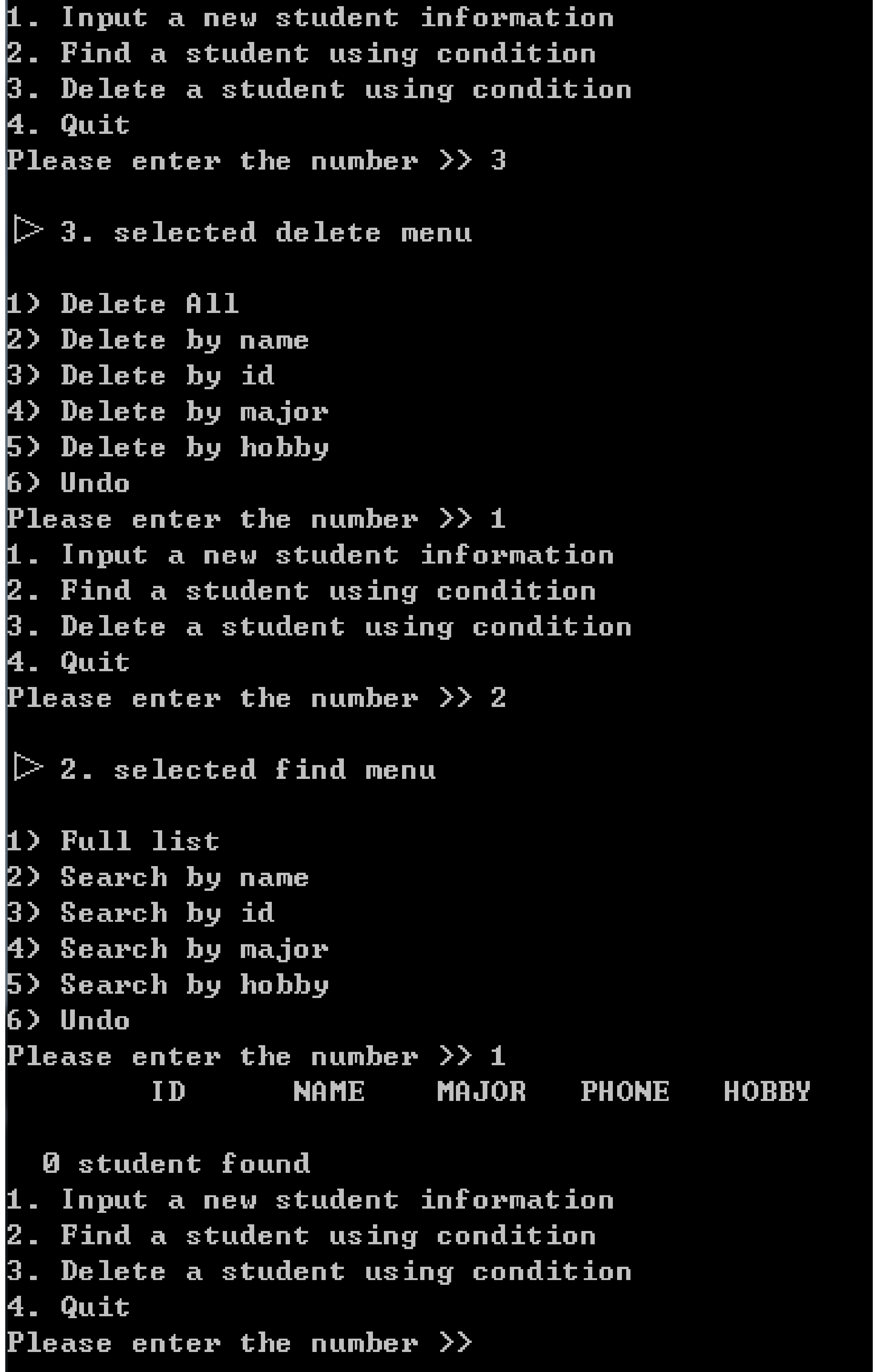
```
Please enter the number >> 1
        ID       NAME    MAJOR   PHONE   HOBBY
2014123456jj        jj        cs  01012345678     sleep

  1 student found
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >>
```



```
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >> 2

▷ 2. selected find menu

1) Full list
2) Search by name
3) Search by id
4) Search by major
5) Search by hobby
6) Undo
Please enter the number >> 2
Name >> james
        ID       NAME    MAJOR   PHONE   HOBBY

  0 student found
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >> 2

▷ 2. selected find menu

1) Full list
2) Search by name
3) Search by id
4) Search by major
5) Search by hobby
6) Undo
Please enter the number >> 2
Name >> jj
        ID       NAME    MAJOR   PHONE   HOBBY
2014123456          jj        cs  01012345678     sleep

  1 student found
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >>
```



```
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >> 3

▷ 3. selected delete menu

1) Delete All
2) Delete by name
3) Delete by id
4) Delete by major
5) Delete by hobby
6) Undo
Please enter the number >>
```



```
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >> 3

▷ 3. selected delete menu

1) Delete All
2) Delete by name
3) Delete by id
4) Delete by major
5) Delete by hobby
6) Undo
Please enter the number >> 1
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >> 2

▷ 2. selected find menu

1) Full list
2) Search by name
3) Search by id
4) Search by major
5) Search by hobby
6) Undo
Please enter the number >> 1
        ID      NAME    MAJOR   PHONE   HOBBY

  0 student found
1. Input a new student information
2. Find a student using condition
3. Delete a student using condition
4. Quit
Please enter the number >>
```

Homework form

- Homework submission e-mail:

hizorro99@naver.com

- E-mail title: day(Thursday or Friday)_name_#week
  - Ex) Friday_james_week13
  - Ex) 목요일반_장원철_13주차
- File title: student id_name_#.c
  - Ex) 2014123456_james_22.c (or .cpp)