C Programming

Practice 10

Structures and ADTs (Abstract Data Types)

Recursion

```c
#include <stdio.h>
void count_down(int n);
int main(void)
{
    count_down(10);
    return 0;
}
void count_down(int n)
{
    if (n == 0)
        printf("₩n*** BLAST OFF ***₩n");
    else if (n > 0) {
        printf("%d !  ", n);
        count_down(n - 1);
    }
    printf("Exiting from count_down...₩n");
}
```

Homework20 - factorial

- Input a number
- Using the recursion
- Print factorial value of the number

```
input the number : 15
15!  : 2004310016
```

The structure Type

```
struct Man { // Different data type !
    char name[30];
    int student_num[13];
    int tel[20];
    char addr[50];
};

struct Man member = { ... };
```

The structure Type + typedef

```
enum suit  { club = 1, diamond = 2, heart = 3, spade = 4 };
typedef enum suit suit;

struct card {
    int pips;
    suit  suit_name;
};
typedef struct card  card;

card  c1 = { 5, spade }, c2;
c2.pips = 12;  /* use of the member operator  "."  */
c2.suit_name = diamond;
```

Accessing a Member

```
#include <stdio.h>
#define  CLASS_SIZE  100
struct student  {
    char   *last_name;
    int   student_id;
    char    grade;
};
```

Using member operator “.”

student class[CLASS_SIZE];

class[0].student_id = 222111;

class[0].grade = ‘A’;

Using pointer notation “->”

student class[CLASS_SIZE];

student *p = class;

p->student_id = 222111;

p->grade = ‘A’;

Structures, Functions, and Assignment(1)

```
struct card {
    int pips;
    char suit;
};
void assign_values(struct card *c_ptr, int p, char s)
{
    c_ptr->pips = p;
    c_ptr->suit = s;
}
void extract_values(struct card *c_ptr, int *p_ptr, char *s_ptr)
{
    *p_ptr = c_ptr->pips;
    *s_ptr = c_ptr->suit;
}
```

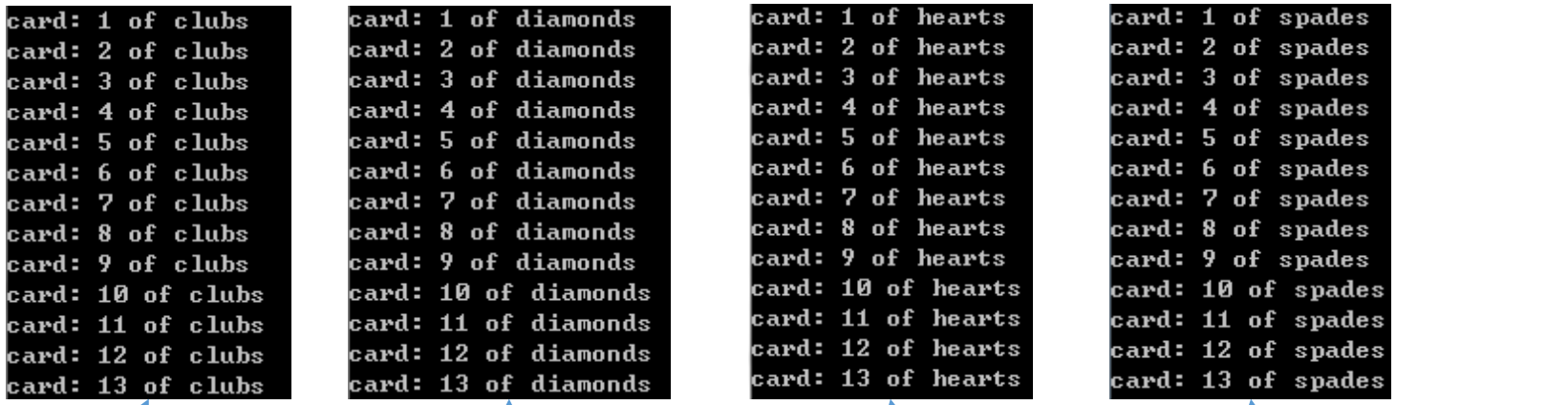
Structures, Functions, and Assignment(2)

```
void prn_values(struct card *c_ptr)
{
    int p;
    char s;
    char *suit_name;
    extract_values(c_ptr, &p, &s);
    switch (s){
        case 'c':
            suit_name = "clubs";
            break;
        case 'd':
            suit_name = "diamonds";
            break;
        case 'h':
            suit_name = "hearts";
            break;
        case 's':
            suit_name = "spades";
            break;
        default:
            suit_name = "error";
    }
    printf("card: %d of %s₩n", p, suit_name);
}
```

Structures, Functions, and Assignment(3)

```
int main(void)
{
    struct card  deck[52];   /* deck of cards */
    int i;
    /* initialize the deck of cards */
    for (i = 0; i< 13; ++i) {
        assign_values(deck + i, i + 1, 'c');
        assign_values(deck + i + 13, i + 1, 'd');
        assign_values(deck + i + 26, i + 1, 'h');
        assign_values(deck + i + 39, i + 1, 's');
    }
    /* print out the hearts */
    for (i = 0; i< 13; ++i)
        prn_values(deck + i + 26);  /* prn_values(&deck[i+26]) */
    return 0;
}
```

Structures, Functions, and Assignment(4)




```
prn_values(deck + i);
// prn_values(&deck[i])
```

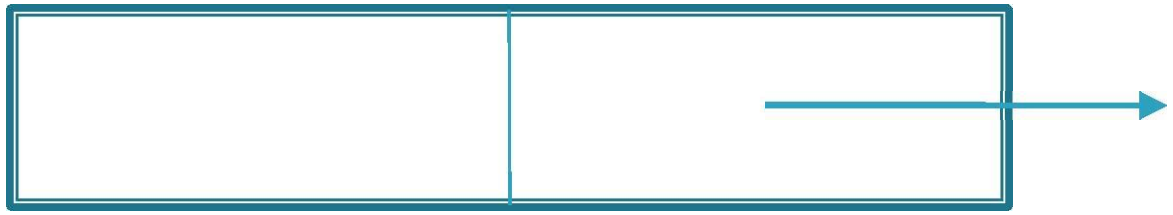
```
prn_values(deck + i + 13);
// prn_values(&deck[i+13])
```

```
prn_values(deck + i + 26);
// prn_values(&deck[i+26])
```

```
prn_values(deck + i + 39);
// prn_values(&deck[i+39])
```

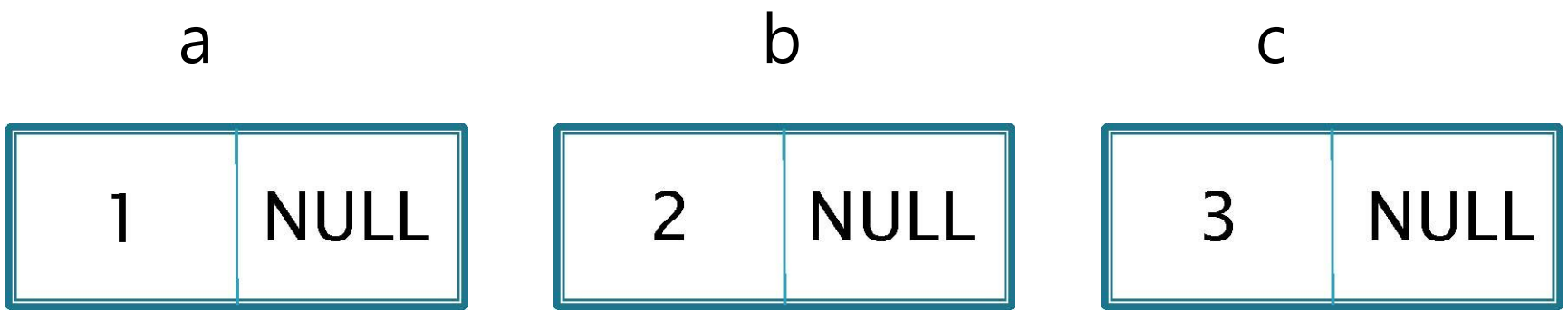
Linked-list

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



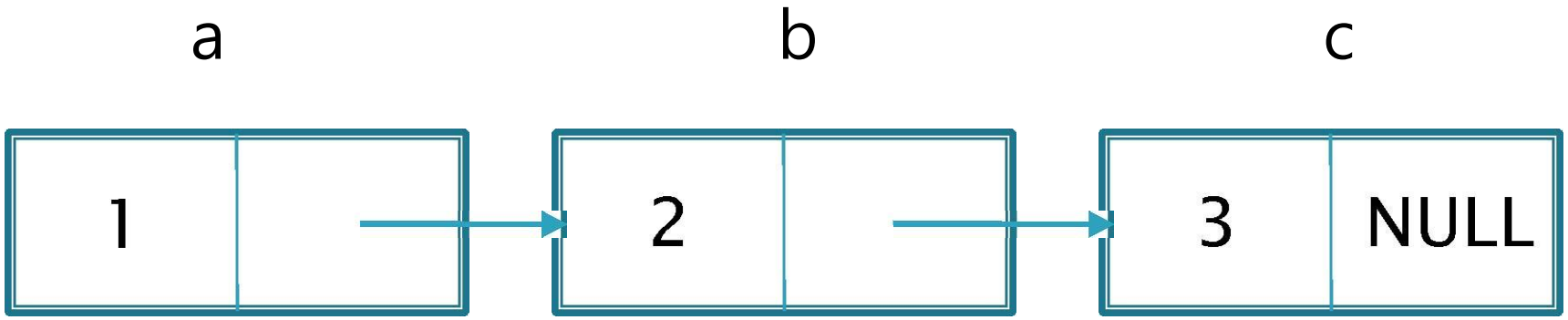
Linked-list

```
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;
    }
}
```

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

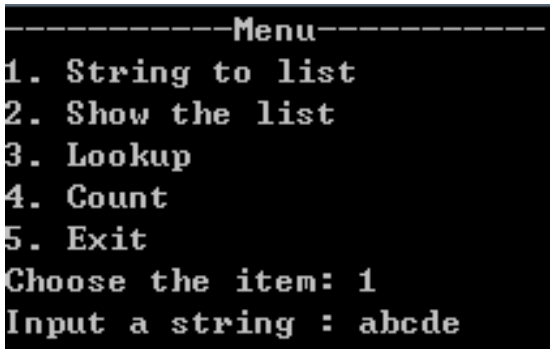
```
/* Insert an element in a linked list:
by having two adjacent elements pointed at by
p1 and p2 and by inserting between them an element pointed at by q.*/
void insert(LINK p1, LINK p2, LINK q)
{
    p1->next = q;  /* insertion */
    q->next = p2;
}
```

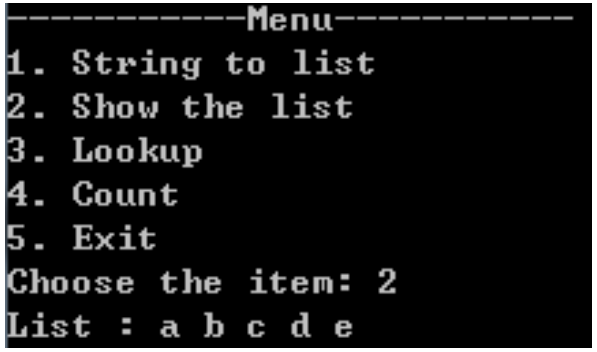
Deleting an element

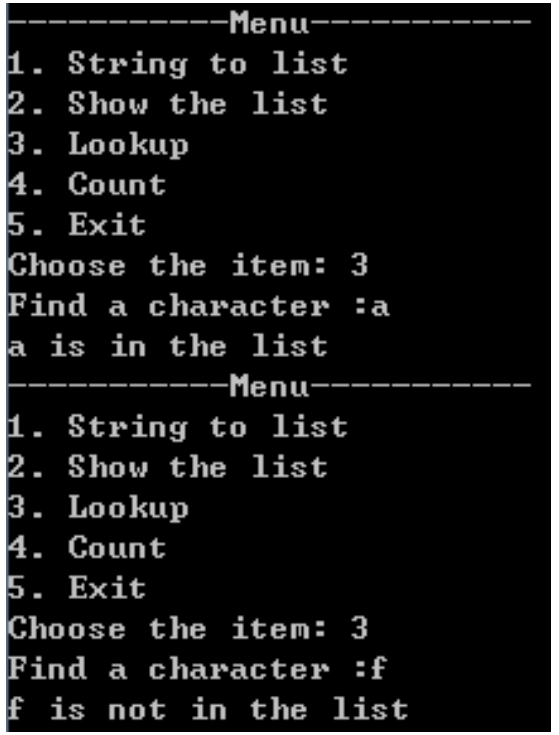
```c
/* Delete a linked list recursively. */
void delete_list(LINK head)
{
    if (head != NULL) {
        delete_list(head->next);
        free(head);   /* release storage */
    }
}
```

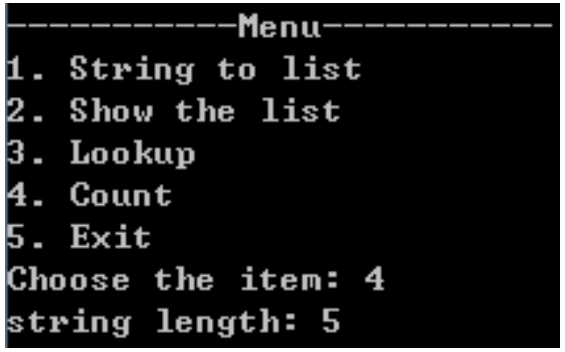
Homework 21 – Linear Linked Lists

- Write a program using the given functions
- Must use a structure type









Homework form

- Homework submission e-mail:

hizorro99@naver.com

- E-mail title: day(Thursday or Friday)_name_#week
  - Ex) Friday_james_week12
  - Ex) 목요일반_장원철_12주차
- File title: student id_name_#.c
  - Ex) 2014123456_james_20.c (or .cpp)
  - Ex) 2014123456_james_21.c (or .cpp)