Programming Assignment #1

2020 Data Structure

제출 : 주석을 포함한 코드 3 개(문제 1, 2 ,3)를 git 에 제출

제출 기한 : ~ 2020/05/22 23:59

1. 개인 실습의 AVL tree 코드에서 특정 키 값을 가진 노드를 삭제하는 delete 함수를 구현하여 추가하라.

- 초기 선언

```
struct avl_node   {
    struct avl_node *left_child, *right_child;  /* Subtrees. */
    int data;                   /* Pointer to data. */
};
```

- 함수

- struct avl_node* avl_delete(struct avl_node *root, int new_key)

  -> 특정 키 값을 가진 노드를 삭제하는 함수. 구현하여야함.

- void display -> 트리 출력 함수

- struct avl_node* avl_add(struct avl_node **root, int new_key) -> 삽입 함수

- struct avl_node* rebalance(struct avl_node **node) -> 균형을 위해 재배치하는 함수

- int get_height_diff(struct avl_node *node) -> 균형 인수를 구하는 함수

- int get_height(struct avl_node *node) -> 트리의 높이를 구하는 함수

- struct avl_node* rotate_left_right(struct avl_node *parent) -> 왼쪽-오른쪽 회전 함수

- struct avl_node* rotate_right_left(struct avl_node *parent) -> 오른쪽-왼쪽 회전 함수

- struct avl_node* rotate_left(struct avl_node *parent) -> 왼쪽 회전 함수

- struct avl_node* rotate_right(struct avl_node *parent) -> 오른쪽 회전 함수

2. 다항식들을 표현하고 관리하는 연결 할당 시스템을 설계한다. 헤드 노드를 가진 원형 연결 리스트를 사용하고, 다항식의 각 항은 다음 구조를 가진 노드로 표현한다. (연결리스트 구조는 FUNDAMENTALS OF DATA STRUCTURES IN C. 2nd Edition. 4.4.4 의 figure 4.15의 구조를 따른다.)

coef	expon	link

해당 시스템은 다음의 기능들을 보유하고 있어야 한다.

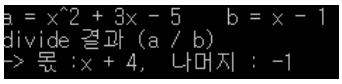
(1) read. 다항식을 읽고 그것을 원형 표현으로 변환하고, 다항식의 헤드 노드의 포인터를 반환

(2) display. 다항식을 출력

(3) mult. 다항식 a와 b에 대하여, a * b를 계산하여 결과를 출력

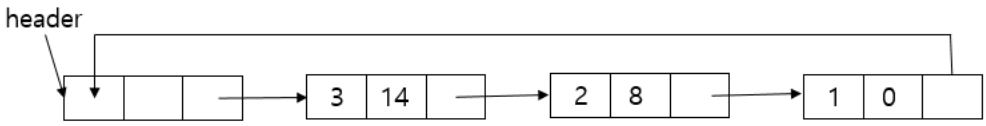
(4) divide. 다항식 a와 b에 대하여 a / b를 계산하여 결과를 출력 (몫과 나머지를 구한다.)

-> 예시



(6) eval. 다항식을 함수 f(x)로 생각하고, 정수 값 c에 대하여 f(c)의 값을 출력

- 표현 예시



3x^14 + 2x^8 + 1

- 초기 선언

```
// Structure of Polynomial List
typedef struct _poly_node *poly_pointer;
typedef struct _poly_node
{
    int coef;
    int expon;
    poly_pointer link;
} poly_node;
```

- coef : 항의 계수

- expon : 항의 지수

- link : 다음 항(다음 노드)에 대한 포인터

- 함수

  - poly_pointer PolynomialRead();

    -> 다항식을 입력 받아 원형 연결 리스트로 변환하는 함수

  - void PolynomialWrite(poly_pointer polynomial);

    -> 입력한 다항식을 출력하는 함수

  - poly_pointer mult(poly_pointer first, poly_pointer second);

    -> 두 다항식 a(first), b(second)의 곱을 구하여 반환하는 함수

  - int divide(poly_pointer first, poly_pointer second, poly_pointer quotient, poly_pointer remainder);

    -> 두 다항식 a(first), b(second)에 대하여 a / b 를 구하는 함수 (몫과 나머지를 구함)

    -> quotient 와 remainder 는 몫과 나머지를 표현한 리스트의 헤더포인터. 함수 호출 전에 NULL 로 초기화하고, 호출 시 parameter 로 사용해야함

    -> 실행 결과를 return 함. (성공 : 1, 실패 : 0)

  - void eval(poly_pointer polynomial, double f);

    -> 어떤 실수 값에 대해 다항식을 계산하여 출력하는 함수

3. 연결 리스트 표현을 사용하여 희소 행렬에 대해 산술 연산을 수행할 수 있는 연결 리스트 시스템을 구현하고자 한다. 메뉴 방식으로 다음과 같은 연산을 수행하는 시스템을 개발하라. (연결리스트 구조는 FUNDAMENTALS OF DATA STRUCTURES IN C. 2nd Edition. 의 181 page의 구조를 따른다.) (또한, 희소 행렬을 읽어 오고 출력하는 함수는 책에 이미 구현이 되어 있는 상태이다.)

(1) madd. 희소 행렬 d = a + b를 계산

(2) mmult. 희소 행렬 d = a * b를 계산

(3) mtranspose. 희소 행렬의 전치 행렬을 구함

- 초기 선언

```
typedef enum
{
    head,
    entry
} tagfield;
typedef struct matrix_node *matrix_pointer;
```

```
typedef struct entry_node
{
    int row;
    int col;
    int value;
} entry_node;
```

- entry_node는 희소 행렬의 원소 값(행, 열, 값)을 나타낸다.

```
typedef struct matrix_node
{
    matrix_pointer down;
    matrix_pointer right;
    tagfield tag;
    union {
        matrix_pointer next;
        entry_node entry;
    } u;
} matrix_node;
```

- 행렬 리스트의 구조체

    - head 노드와 entry 노드를 나타내기 위한 tag필드가 있음

    - down : column(열)의 노드들을 리스트로 연결할 때 사용

    - right : row(행)의 노드들을 리스트로 연결할 때 사용

    - next : 헤드 노드들을 연결할 때 사용

- 함수

  - matrix_pointer new_node(void);

    -> 새 행렬 노드를 생성하여 반환하는 함수

  - matrix_pointer madd(matrix_pointer m1, matrix_pointer m2);

    -> 두 행렬을 더하여 반환하는 함수

  - matrix_pointer mtranspose(matrix_pointer node);

    -> 행렬을 전치 행렬로 변환하는 함수

  - matrix_pointer mmult(matrix_pointer m1, matrix_pointer m2);

    -> 두 행렬의 곱을 구하여 반환하는 함수

  - void printf_menu();

    -> 메인 메뉴 출력 함수

- 입력 예시

```
-----------------------------
      M    E    N    U
1. Matrix Read
2. Matrix Write
3. Matrix Erase
4. Matrix Add
5. Matrix Multiply
6. Matrix Transpose
7. Program Exit
-----------------------------
You select a menu number
Menu: 1
-------------------------
첫 번째 희소행렬을 입력하세요.
Enter the number of rows, columns and number of nonzero terms: 3 3 3
Enter row, column and value: 0 1 2
Enter row, column and value: 0 2 1
Enter row, column and value: 2 2 3

두 번째 희소행렬을 입력하세요.
Enter the number of rows, columns and number of nonzero terms: 3 3 2
Enter row, column and value: 0 2 2
Enter row, column and value: 1 0 1
```

➔ 입력된 행렬

	0	1	2
0	0	2	1
1	0	0	0
2	0	0	3

	0	1	2
0	0	0	2
1	1	0	0
2	0	0	0

```
Menu: 2
--------------------------
첫 번 째  입 력 한  희 소  행 렬  출 력 ..

 num_rows = 3, num_cols = 3
The matrix by row, column, and value:

    0    1    2
    0    2    1
    2    2    3

두 번 째  입 력 한  희 소  행 렬  출 력 ..

 num_rows = 3, num_cols = 3
The matrix by row, column, and value:

    0    2    2
    1    0    1
```

```
Menu: 4
--------------------------------
입 력 한  두 개 의  희 소  행 렬  더 하 기 ..

 num_rows = 3, num_cols = 3
The matrix by row, column, and value:

    0    1    2
    0    2    3
    1    0    1
    2    2    3
--------------------------------
```

```
Menu: 5
--------------------------------
입 력 한  두 개 의  희 소  행 렬  곱 하 기 ..

 num_rows = 3, num_cols = 3
The matrix by row, column, and value:

    0    0    2
```

```
Menu: 6
-----------------------------------
입 력 한  희 소  행 렬 을  전 치  행 렬 로  변 환 .

 num_rows = 3, num_cols = 3
The matrix by row, column, and value:

    1    0    2
    2    0    1
    2    2    3

 num_rows = 3, num_cols = 3
The matrix by row, column, and value:

    2    0    2
    0    1    1
```