자료구조응용

1. 다음과 같이 헤더노드를 가진 단일 환형연결리스트 (singly linked circular list)을 이용한 다항식 더하기 프로그램을 작성하라.

(1) 함수정의

<교재 함수 수정하여 구현>

① insertFront2

Program 4.18 insertFront를 수정. 입력되는 순서대로 리스트의 처음, 즉 헤더노드 바로 다음에 추가된다. 노드 추가 시 getNode() 호출

② insertLast

Program 4.18 insertFront를 수정. 입력되는 순서대로 리스트의 마지막에 추가된다. insertFront 함수의 else 블록 마지막에 *last = node; 추가함으로써 간단히 구현됨.
노드 추가 시 getNode() 호출

③ inputPolyCL

파일로부터 "헤더노드를 가진 단일 환형연결리스트"로 된 다항식을 생성함

④ printCList

Program 4.19를 수정. 헤더노드를 제외한 항들만 출력되도록 할 것.

⑤ attach

Program 4.10를 수정. MALLOC(temp, sizeof(*temp)); 대신에 temp = getNode();를 사용

<교재 함수 그대로 구현 : Program 4.11 ~ 4. 15>

① erase, ② getNode, ③ retNode, ④ cerase, ⑤ cpadd

(2) 실행 순서

① 두 개의 입력받은 다항식 a, b를 각각 헤더노드를 가진 단일 환형연결리스트 형태로 구현하고 last 포인터를 유지한다. ※ inputPolyCL 2회 호출

※ $a = 3x^{14} + 2x^8 + 1$, $b = 8x^{14} - 3x^{10} + 10x^6$ 에 대한 입력 예

[입력]

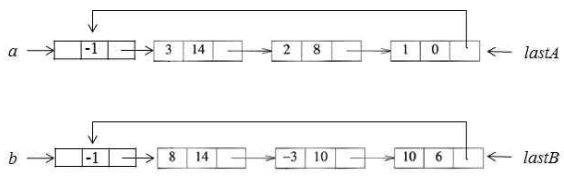
a 1 0 2 8 3 14 //a 입력 (계수/지수 순으로 입력)

d 8 14 -3 10 10 6 //b 입력

or

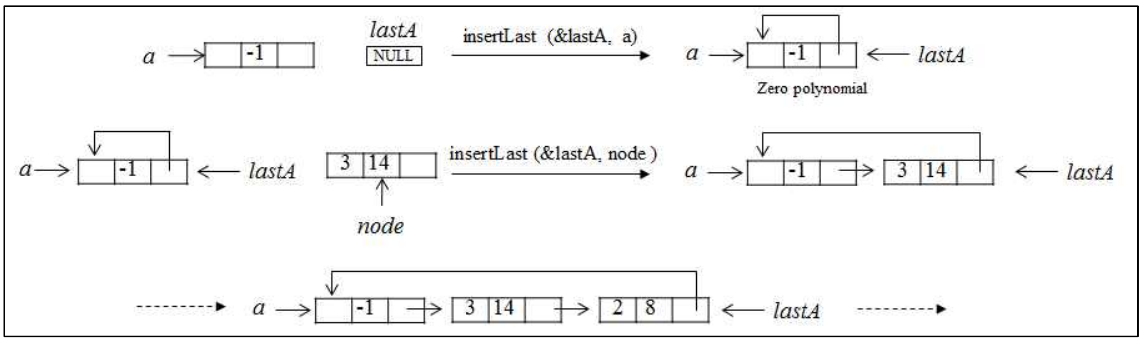
d 3 14 2 8 1 0 //a 입력

a 10 6 -3 10 8 14 //b 입력

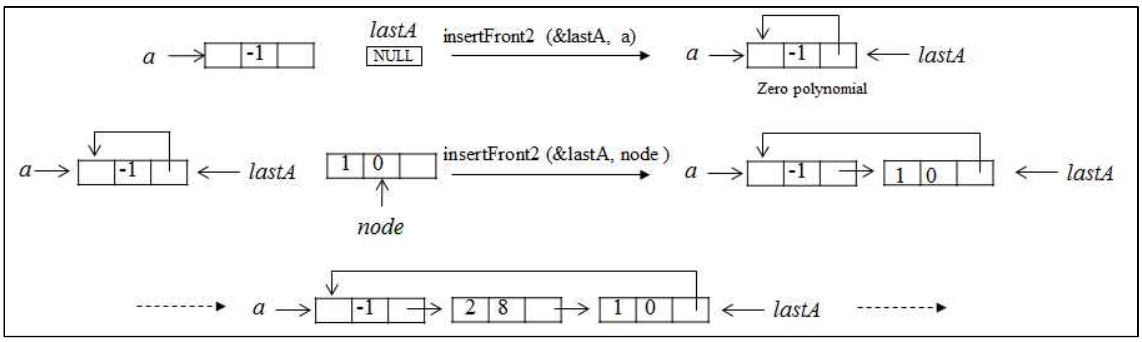


※ 첫 문자 입력이 'a'이면 지수 차수에 대해 오름차순(ascending order), 'd'이면 내림차순(descending order)으로 입력됨. **오름차순으로 입력되면 각 노드는 환형리스트의 첫 노드로** 삽입되어야 하며, **내림차순으로 입력되면 각 노드는 환형리스트의 마지막 노드로** 추가

<내림차순 입력에 대한 다항식 생성($a = 3x^{14} + 2x^8 + 1$)>



<오름차순 입력에 대한 다항식 생성($a = 3x^{14} + 2x^{8} + 1$)>



※ 주의: 위 그림의 경우라면 첫 노드(1, 0) 삽입 시 last를 변경하고 이후는 변경 없음

② a, b 두 다항식의 정보를 출력한다. ※ printCList
③ a+b의 결과를 c에 저장하는 다항식 더하기를 수행한다. ※ cpadd
④ 다항식 c를 출력한다. ※printCList
⑤ 다항식 a, b, c를 ***avail*** 에 반납한다. ※ cerase
⑥ avail을 삭제한다. ※ erase

(3) 구현 세부사항

※ 주의 : a, b, c는 헤더노드를 가진 단일 환형연결리스트이며, avail은 단일연결리스트임

```
typedef struct polyNode *polyPointer;
typedef struct polyNode {
        int coef;
        int expon;
        polyPointer link;
        } polyNode;
polyPointer a,b;

polyPointer c, lastA, lastB, avail = NULL;
```

coef	expon	link

```
polyPointer cpadd(polyPointer a, polyPointer b)
{/* polynomials a and b are singly linked circular lists
    with a header node. Return a polynomial which is
    the sum of a and b */
  polyPointer startA, c, lastC;
  int sum, done = FALSE;
  startA = a;          /* record start of a */
  a = a→link;          /* skip header node for a and b*/
  b = b→link;
  c = getNode();       /* get a header node for sum */
  c→expon = -1; lastC = c;
  do {
     switch (COMPARE(a→expon, b→expon)) {
        case -1: /* a→expon < b→expon */
              attach(b→coef,b→expon,&lastC);
              b = b→link;
              break;
        case 0:  /* a→expon = b→expon */
              if (startA == a)  done = TRUE;
              else {
                 sum = a→coef + b→coef;
                 if (sum) attach(sum,a→expon,&lastC);
                 a = a→link; b = b→link;
              }
              break;
        case 1:  /* a→expon > b→expon */
              attach(a→coef,a→expon,&lastC);
              a = a→link;
     }
  } while (!done);
  lastC→link = c;
  return c;
}
```

Program 4.15: Adding two polynomials represented as circular lists with header nodes

※ 기타 함수는 교재 및 강의자료 참고

(3) 실행 예

a 1 0 2 8 3 14 //자동 입력

d 8 14 -3 10 10 6 //자동 입력

11 14 -3 10 2 8 10 6 1 0 //결과 출력

or

d 3 14 2 8 1 0 //자동 입력

a 10 6 -3 10 8 14 //자동 입력

11 14 -3 10 2 8 10 6 1 0 //결과 출력

2. **다음과 같이 집합 S에 대한 동치관계(equivalence relation, ≡)가 성립할 때 S의 동치류(equivalence class)를 구하는 프로그램을 작성하라. 연결리스트를 이용하여야 한다.**

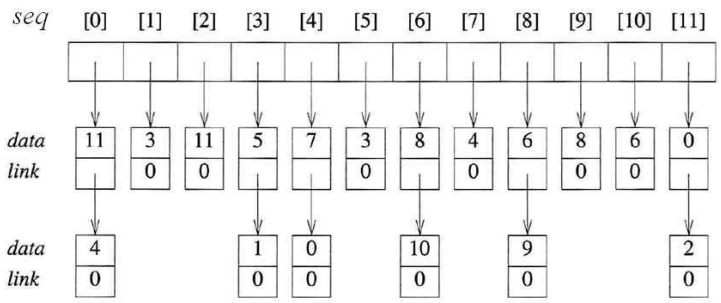
S = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

0≡4, 3≡1, 6≡10, 8≡9, 7≡4, 6≡8, 3≡5, 2≡11, 11≡0

(1) 실행 순서

① 파일입력을 통해 집합 S의 크기와 동치관계를 나타내는 순서쌍 데이터를 입력받으면서 자료구조를 생성한다.

입력
12 //s의 크기
9 //동치관계의 수
0 4 //동치관계1
3 1
6 10
8 9
7 4
6 8
3 5
2 11
11 0 //동치관계 끝



② 동치류를 구하여 출력한다.

(2) 구현세부사항

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 24
#define FALSE 0
#define TRUE 1
typedef struct node *nodePointer;
typedef struct node {
        int data;
        nodePointer link;
        } node;
void main(void)
{
   short int out[MAX_SIZE];
   nodePointer seq[MAX_SIZE];
   nodePointer x,y,top;
   int i,j,n;

   printf("Enter the size (<= %d) ",MAX_SIZE);
   scanf("%d",&n);
   for (i = 0; i < n; i++) {
   /* initialize seq and out */
      out[i] = FALSE ;  seq[i] = NULL;
   }

   /* Phase 1: Input the equivalence pairs: */
   printf("Enter a pair of numbers (-1 -1 to quit): ");
   scanf("%d%d",&i,&j);
   while (i >= 0) {
      MALLOC(x, sizeof(*x));
      x→data = j;  x→link = seq[i];  seq[i] = x;
      MALLOC(x, sizeof(*x));
      x→data = i;  x→link = seq[j];  seq[j] = x;
      printf("Enter a pair of numbers (-1 -1 to quit): ");
      scanf("%d%d",&i,&j);
   }
```

```
    /* Phase 2: output the equivalence classes */
    for (i = 0; i < n; i++)
       if (out[i] == FALSE){
          printf("\nNew class: %5d",i);
          out[i] = TRUE;    /* set class to true  */
          x = seq[i];  top = NULL; /* initialize stack */
          for (;;) {      /* find rest of class */
             while (x) {  /* process list */
                j = x→data;
                if (out[j] == FALSE){
                   printf("%5d",j);  out[j] = TRUE;
                   y = x→link; x→link = top; top = x; x = y;
                }
                else x = x→link;
             }
             if (!top) break;
             x = seq[top→data]; top = top→link;
                                          /* unstack */
          }
       }
}
```

Program 4.22: Program to find equivalence classes

(3) 실행 예

```
12 //s의 크기
9 //동치관계의 수
0 4 //동치관계1
3 1
6 10
8 9
7 4
6 8
3 5
2 11
11 0 //동치관계 끝
0 2 4 7 11 //new class sorting
1 3 5 //new class sorting
6 8 9 10 //new class sorting
```

3. **다음과 같이 정수 데이터를 입력하면서 "헤더노드를 가진 이중연결환형리스트 (doubly linked circular list)"를 만들고 실행예와 같이 수행되는 프로그램을 작성하라.**

(1) 실행 순서

① 다음 입력으로 부터 데이터를 입력받는 순서대로 이중환형연결리스트의 마지막 노드로 추가 되도록 한다.

50 80 30 20 19 90 30 55 77 30
99 45 55 89 91 10 20 66 38 59
22 55 88 22 66 29 50 95 78 83

② 순방향과 역방향으로 노드의 데이터를 출력한다. (forward & backward)

③ 성적이 입력 점수 이하인 노드를 삭제한다.

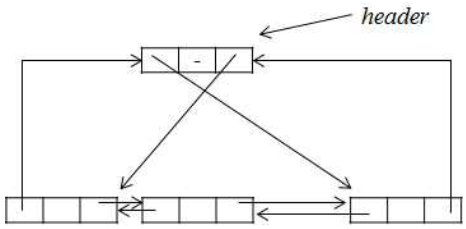
④ 순방향과 역방향으로 노드의 데이터를 출력한다.

⑤ 헤더를 제외한 모든 노드를 삭제한다.

(2) 구현 세부사항

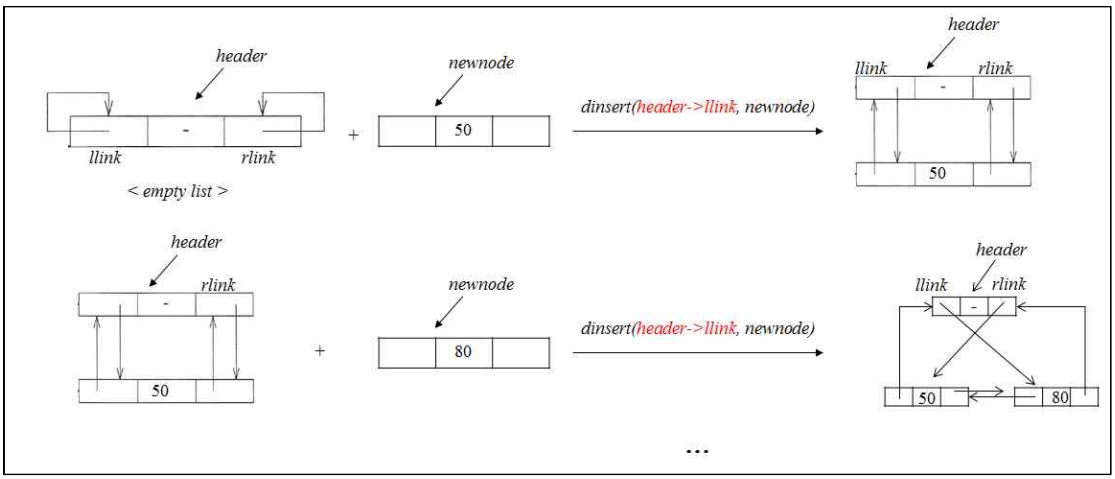
① 구조체 선언문

```
typedef struct node *nodePointer;
typedef struct node {
        nodePointer llink;
        int data;
        nodePointer rlink;
        } node;
nodePointer header = NULL;
```



② 이중연결환형리스트의 Last node로 추가하기

- empty list를 생성한 후 새로운 노드를 하나씩 추가해 간다.
- header node의 llink는 last node pointer 역할을 한다. (※ dinsert 함수 호출)



③ 함수정의

```
void dinsert(nodePointer node, nodePointer newnode)
{/* insert newnode to the right of node */
   newnode→llink = node;
   newnode→rlink = node→rlink;
   node→rlink→llink = newnode;
   node→rlink = newnode;
}
```

Program 4.26: Insertion into a doubly linked circular list

```
void ddelete(nodePointer node, nodePointer deleted)
{/* delete from the doubly linked list */
   if (node == deleted)
      printf("Deletion of header node not permitted.\n");
   else {
      deleted→llink→rlink = deleted→rlink;
      deleted→rlink→llink = deleted→llink;
      free(deleted);
   }
}
```

Program 4.27: Deletion from a doubly linked circular list

- 기타 함수들을 적절하게 선언하고 사용할 것(자료구조생성, 리스트 출력, 삭제관련함수 등)

(3) 결과물 출력 (각 출력물은 모두 %3d의 형식을 사용하고 공백을 사용하지 마시오.)

50 80 30 20 19 90 30 55 77 30
99 45 55 89 91 10 20 66 38 59
22 55 88 22 66 29 50 95 78 83 //자동입력
50 80 30 20 19 90 30 55 77 30
99 45 55 89 91 10 20 66 38 59
22 55 88 22 66 29 50 95 78 83 //순방향출력
83 78 95 50 29 66 22 88 55 22
59 38 66 20 10 91 89 55 45 99
30 77 55 30 90 19 20 30 80 50 //역방향출력
50 //비교값 입력
80 90 55 77 99 55 89 91 66 59
55 88 66 95 78 83 //순방향 출력
83 78 95 66 88 55 59 66 91 89
55 99 77 55 90 80 //역방향 출력