//第二题

TYPE SQLIST = RECORD

elem: ARRAY[1..max] OF elemtp;

n:0..max

END;

PROC minus(VAR C:SQLIST; A, B:SQLIST; i, j, k, s:integer);

{把属于A，不属于B的元素放入C中}

BEGIN

INITIATE(C); i := 1; j := 1; k := 1; s := 1;

IF A.n = 0

THEN INSERT(C, k, 0);

ELSE[

IF B.n = 0

THEN

FOR i := 1 TO A.n DO

[INSERT(C, k, A[i]); k := k+1;]

]

WHILE i ≤ A.n DO[

FOR j := s TO B.n DO

[IF A[i] = B[j]

THEN [i := i+1;

IF j = s THEN [s := s+1; j := s]

ELSE j := s;

break;]

]

INSERT(C, k, A[i]); k := k+1; i := i+1;]

END;

//3

TYPE SQLIST = RECORD

elem: ARRAY[1..max] OF elemtp;

n:0..max

END;

PROC invert(VAR i, j:integer; A:SQLIST);

{把每项的第i个元素和第 max - i + 1 个元素替换位置}

BEGIN

i := 1; j := max;

WHILE (i ≠ j) OR (i ≠ j -1) DO

A[i] <-> A[j]

END;

//4.试写出将一个采用单链表存储的线性表 A 的数据元素进行逆置的算法

TYPE pointer = ↑nodetype;

nodetype = RECORD

elem: elemtp;

next: pointer;

END;

linkisttp = pointer;

PROC invert\_list(VAR la, v1, v2:linkisttp; i:integer);

{a为带表头结点的单链表，la是它的头指针，在空表头后再加入两个空表头}

BEGIN

NEW(v1); NEW(v2);

p = la↑.next;

la↑.next = v1; v1↑.next = p;

v1↑.next = v2; v2↑.next = p;

p = la↑.next; p1 = p↑.next; p2 = p1↑.next;

WHILE p1↑.next ≠ nil DO[

p1↑.elem = p;

p = p1;

p1 = p2;

p2 = p2↑.next;

]

p = la↑.next.next;

dispose(la↑.next); la = p; p = p↑.next;

dispose(la↑.next); la = p;

END;

//5.设 A 和 B 是两个升序有序的循环链表，Pa 和 Pb 分别指向两个表的表头结点，试写出一个将这两个表归并为一个有序的循环链表的算法

TYPE pointer = ↑nodetype;

nodetype = RECORD

elem: elemtp;

next: pointer;

END;

linkisttp = pointer;

PROC merge\_list(VAR la, lb, lc, p1, p1\_2 p2, p2\_2, pc1, pc2, pa, pb:linkisttp;);

{pa和pb是表头结点，la和lb是头指针，a和b的结点比较大小，把较大结点复制到循环链表表c中，大的结点的指针向前移动一位，若相等，复制a或b结点中的信息，指针都向前移动一位}

BEGIN

p1 <- pa↑.next; p2 <- pb↑.next; pc1 <- lc;

REPEAT[

NEW(pc2);

IF p1↑.elem < p2↑.elem

THEN [pc2↑.elem <- p2↑.elem; p2 <- p2↑.next;]

ELSE IF p1↑.elem > p2↑.elem

THEN [pc2↑.elem <- p1↑.elem; p1 <- p1↑.next;]

ELSE IF p1↑.elem = p2↑.elem

THEN [pc2↑.elem <- p1↑.elem; p1 <- p1↑.next;

pc2↑.elem <- p2↑.elem; p2 <- p2↑.next;

]

pc1↑.next = pc2;

pc1↑ <- pc2;

]WHILE (p1 ≠ nil) AND (p2 ≠ nil);

IF (p1 = nil) AND (p2 ≠ nil)[

REPEAT[

NEW(pc2);

pc2↑.elem <- p2↑.elem;

pc1↑.next <- pc2;

pc1 <- pc2;

p2 <- p2↑.next;

]WHILE p2 ≠ nil;

]

ELSE IF (p1 ≠ nil) AND (p2 = nil)[

REPEAT[

NEW(pc2);

pc2↑.elem <- p1↑.elem;

pc1↑.next <- pc2;

pc1 <- pc2;

p1 <- p1↑.next;

]WHILE p1 ≠ nil;

]

pc1 <- nil;

//output

pc1 <- lc↑.next;

REPEAT[

IF pc1↑.next = 0

WRITE(pc1↑.elem);

pc1 <- pc1↑.next;

]WHILE pc1 ≠ nil;

//free

p1 <- pa;

WHILE p1↑.next ≠ nil[

p1\_2 = p1↑.next;

FREE(p1);

p1 <- p1\_2;

]

FREE(p1);

FREE(pa);

p1 = nil;

pa = nil;

la = nil;

p2 <- pa;

WHILE p2↑.next ≠ nil[

p2\_2 = p2↑.next;

FREE(p2);

p2 <- p2\_2;

]

FREE(p2);

FREE(pb);

p2 = nil;

pb = nil;

lb = nil;

END;