B62002Y-01 数据结构 2017 年 3 月 22 日

作业一

主讲教师：金蓓弘 张远航 2015K8009929045

第 1 章 绪论

1.8 (1) $n-1$；(2) $n-1$；(3) $n-1$；(4) $\dfrac{n(n+1)}{2}$；

(5) $1+(1+2)+\cdots+(1+2+\cdots+n)=\dfrac{1}{12}n(n+1)(2n+3)$；

(6) n；(7) $\lfloor\sqrt{n}\rfloor$；(8) 1100.

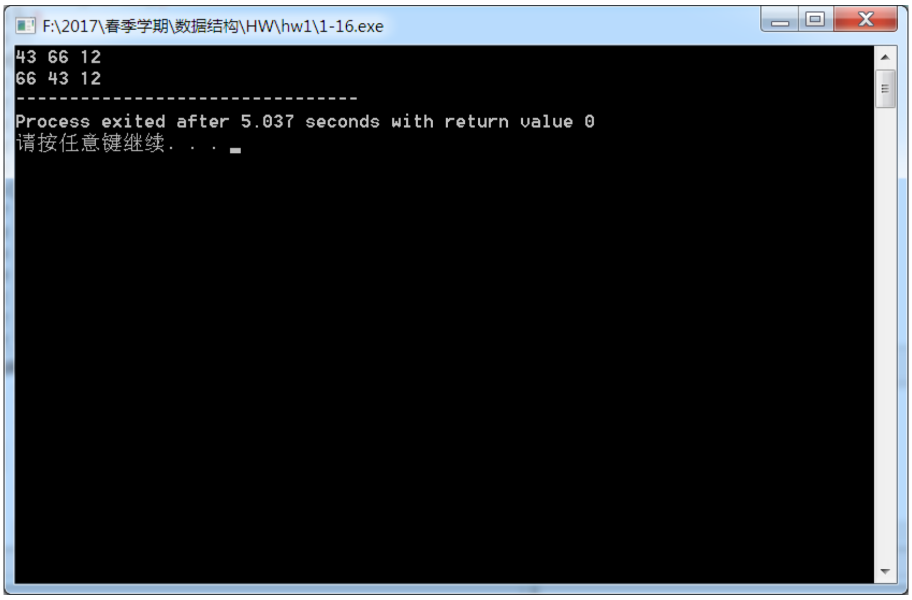
1.9 时间复杂度为 $O(\log_2 n)$，$\texttt{count}=\log_2 n-2$.

1.12 (1) $\surd$；(2) $\times$；(3) $\times$；(4) $\surd$；(5) $\times$.

1.16 代码如下.

```c
#include <stdio.h>

void swap(int *x, int *y){
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

void sort(int *x, int *y, int *z){
    if (*x < *y) swap(x, y);
    if (*x < *z) swap(x, z);
    if (*y < *z) swap(y, z);
}

int main(){
    int x, y, z;
    scanf("%d%d%d", &x, &y, &z);
    sort(&x, &y, &z);
    printf("%d %d %d", x, y, z);

    return 0;
}
```

输入：$\boxed{43\ 66\ 12}$

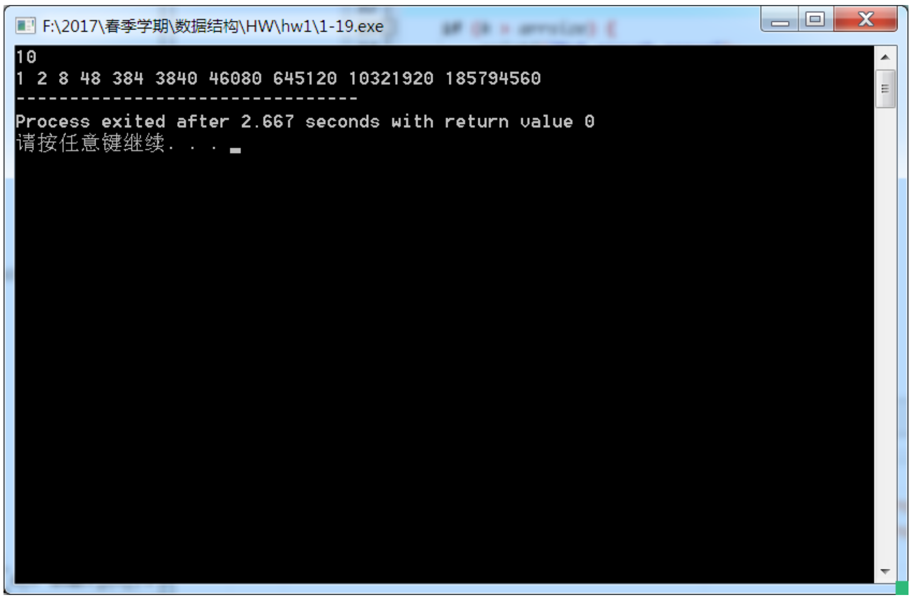

F:\2017\春季学期\数据结构\HW\hw1\1-16.exe

```
43 66 12
66 43 12
--------------------------------
Process exited after 5.037 seconds with return value 0
请按任意键继续. . .
```

1.19 代码如下.

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define MAXINT INT_MAX

const int arrsize = 100;

int main(){
    int i, k, a[arrsize];
    scanf("%d", &k);

    if (k > arrsize) {
        printf("Not enough space");
        exit(0);
    }
    for (i = 0; i < k; i++) {
        if (!i) a[i] = 1;
        else {
            if (a[i-1] > MAXINT / (2*i)) {
                printf("Overflow");
                exit(0);
            }
            a[i] = 2 * i * a[i-1];
        }
    }

    for (i = 0; i < k; i++)
        printf("%d ", a[i]);

    return 0;
}
```

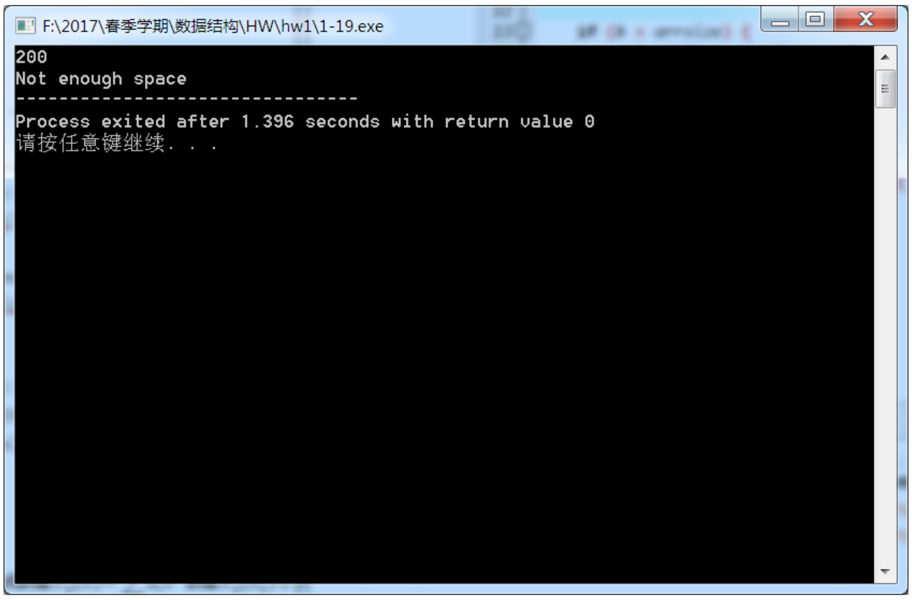
输入：$\boxed{10}$



F:\2017\春季学期\数据结构\HW\hw1\1-19.exe

```
10
1 2 8 48 384 3840 46080 645120 10321920 185794560
--------------------------------
Process exited after 2.667 seconds with return value 0
请按任意键继续. . .
```

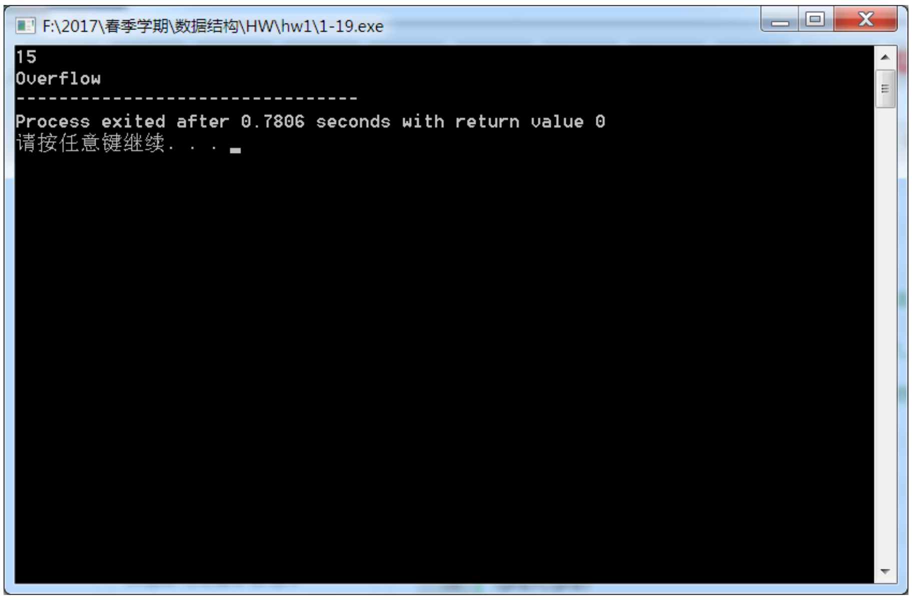
输入：$\boxed{200}$



F:\2017\春季学期\数据结构\HW\hw1\1-19.exe

```
200
Not enough space
--------------------------------
Process exited after 1.396 seconds with return value 0
请按任意键继续. . .
```

输入：$\boxed{15}$

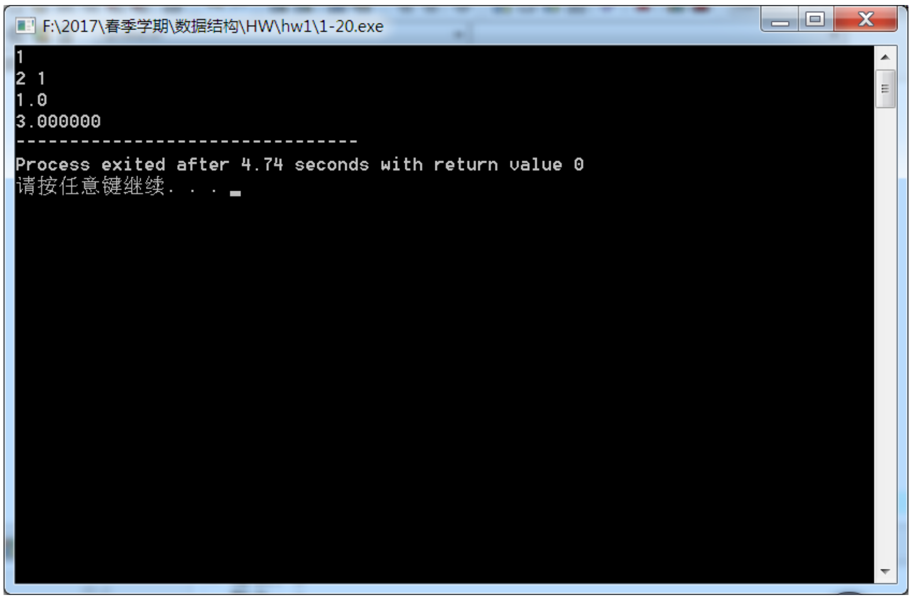


```
15
Overflow
--------------------------------
Process exited after 0.7806 seconds with return value 0
请按任意键继续. . .
```

1.20 代码如下.

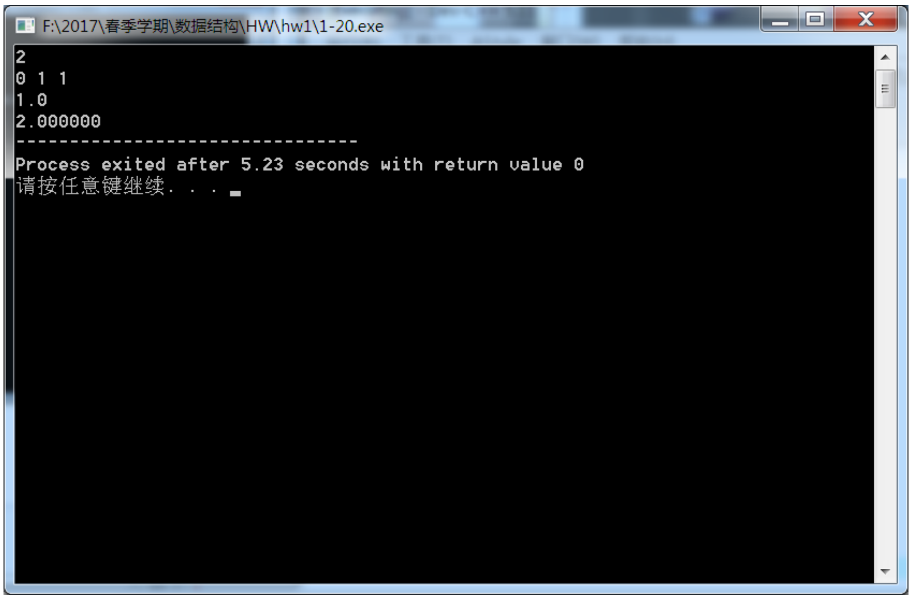
```c
#include <stdio.h>
#define DEGREE 100

double eval(int *a, int n, double x0) {
    int i;
    double ans = (double)*a, power = 1.0;

    for (i = 1; i <= n; i++) {
        power *= x0;
        ans += *(a+i) * power;
    }

    return ans;
}

int main() {
    int n, a[DEGREE];
    double x0;

    int i;
    scanf("%d", &n);
    for (i = 0; i <= n; i++)
        scanf("%d", &a[i]);
    scanf("%lf", &x0);

    printf("%lf", eval(&a[0], n, x0));

    return 0;
}
```

输入函数为 $P_1(x) = x + 2$，取值 $x_0 = 1.0$：



```
1
2 1
1.0
3.000000
--------------------------------
Process exited after 4.74 seconds with return value 0
请按任意键继续. . .
```

输入函数为 $P_2(x) = x^2 + x$，取值 $x_0 = 1.0$：



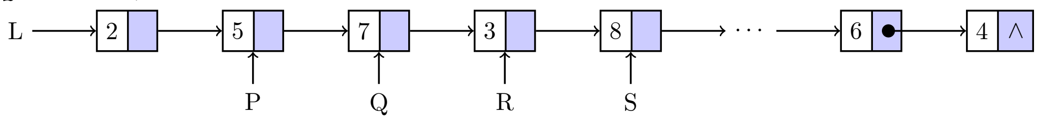
```
2
0 1 1
1.0
2.000000
--------------------------------
Process exited after 5.23 seconds with return value 0
请按任意键继续. . .
```

程序中，`power *= x0;` 和 `ans += *(a+i) * power;` 的执行频度均为 n 次，整个算法的时间复杂度为 $O(n)$.

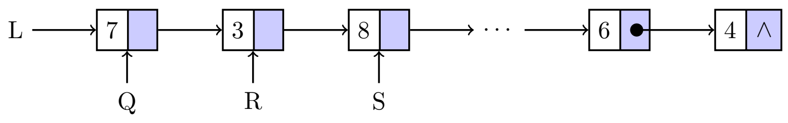
第 2 章 线性表

2.4 示意图如下：

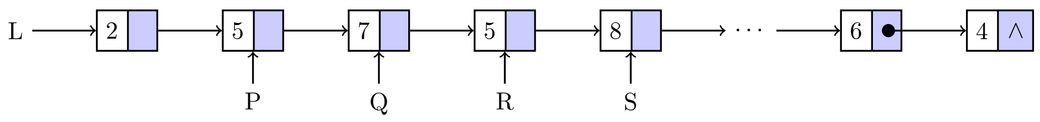
(1) `Q=P->next;`



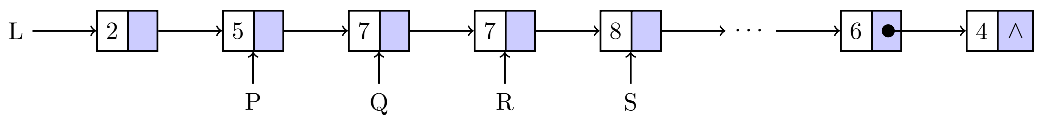
(2) `L=P->next;`



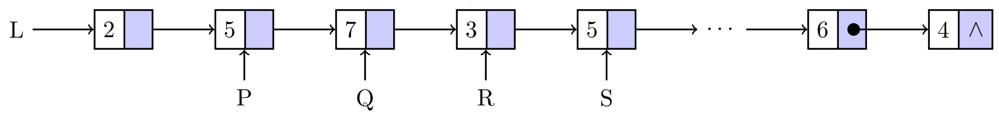
(3) `R->data=P->data;`



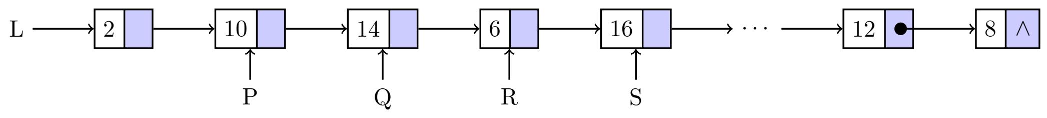
(4) `R->data=P->next->data;`



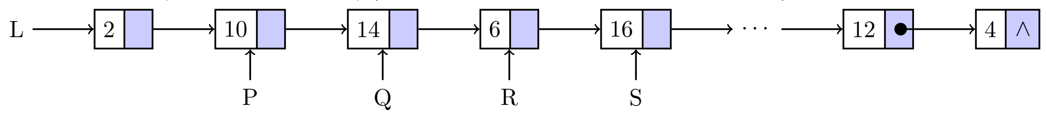
(5) `P->next->next->next->data=P->data;`



(6) `T=P; while (T!=NULL){T->data=T->data*2;T=T->next;}`

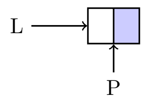


(7) `T=P; while (T->next!=NULL){T->data=T->data*2;T=T->next;}`

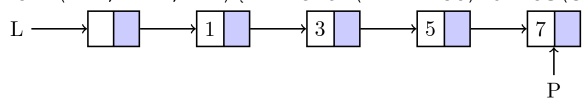


2.5 示意图如下：

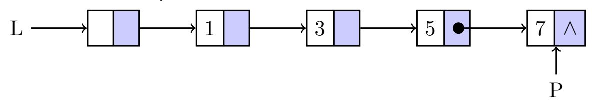
```
L=(LinkList)malloc(sizeof(LNode)); P=L;
```



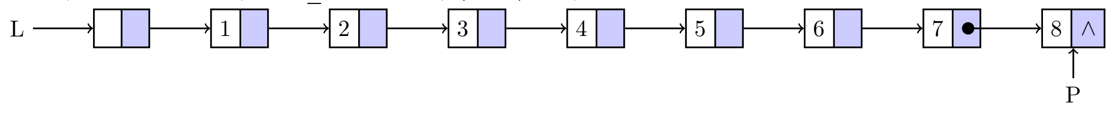
```
for (i=1;i<=4;i++){P->next=(LinkList)malloc(sizeof(LNode)); P=P->next; P->data=i*2-1;}
```



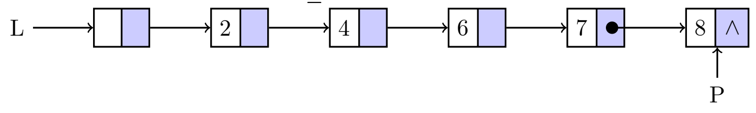
```
P->next=NULL;
```



```
for (i=4;i>=1;i--) Ins_LinkList(L,i+1,i*2);
```



```
for (i=1;i<=3;i++) Del_LinkList(L,i);
```



2.9 (1) 若表 L 的长度不小于 2，将 L 的首结点变成尾结点；

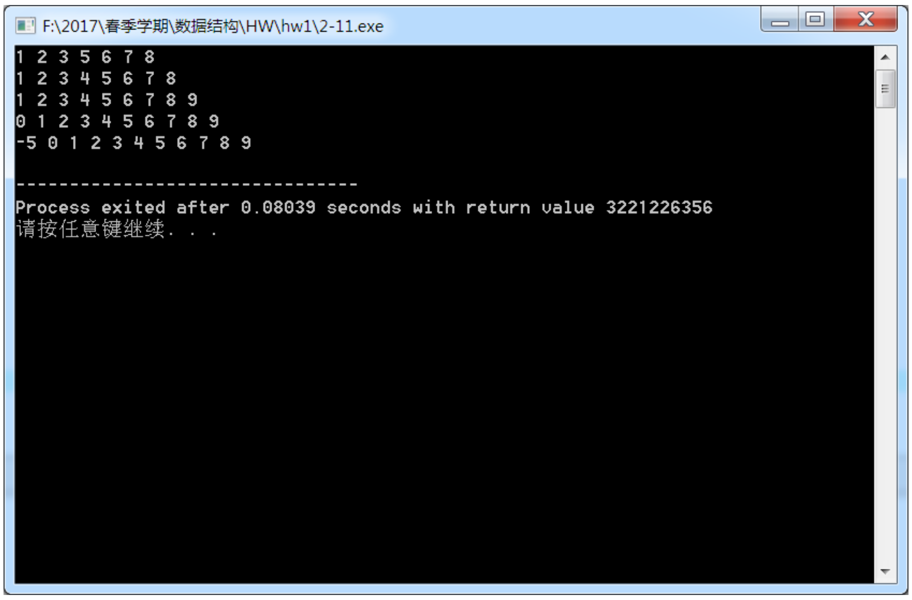
(2) BB 把单循环链表中的两个结点直接连接起来，AA 利用 BB 将一个单循环链表拆成两个单循环链表.

2.11 代码如下.

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

#define OK 1
#define ERROR 0
#define OVERFLOW -2

#define LIST_INIT_SIZE 100
#define LISTINCREMENT 10

// assume the elements are of type int
typedef int ElemType;
typedef int Status;

typedef struct {
    ElemType *elem;
    int length;
    int listsize;
} SqList;

int precede(ElemType x, ElemType y) {
    return x < y;
}

Status InitList_Sq(SqList *L) {
    L -> elem = (ElemType*)malloc(LIST_INIT_SIZE * sizeof(ElemType));
    if (!L -> elem) exit(OVERFLOW);
    L -> length = 0;
    L -> listsize = LIST_INIT_SIZE;

    return OK;
}
```

```

// insert x into the non-decreasing list va while maintaining order
Status InsertElem_Sq(SqList *va, ElemType x) {
    if (va -> length == va -> listsize) {
        ElemType *newbase = (ElemType*)realloc(va -> elem, (va -> listsize +
            LISTINCREMENT) * sizeof(ElemType));
        if (!newbase) return ERROR;
        va -> elem = newbase;
        va -> listsize += LISTINCREMENT;
    }

    int i;
    for (i = va -> length; (i > 0) && precede(x, va -> elem[i-1]); i--)
        va -> elem[i] = va -> elem[i-1];
    va -> elem[i] = x;
    ++va -> length;

    return OK;
}

void PrintList_Sq(SqList *L) {
    int i;

    for(i = 0; i < L -> length; i++)
        printf("%d ", L -> elem[i]);
    printf("\n");
}

void DestroyList_Sq(SqList *L) {
    free(L);
}

// test case
int main(){
    SqList va;
    int a[] = {1, 2, 3, 5, 6, 7, 8};

    if (InitList_Sq(&va) != ERROR) {
        int i;
        for (i = 0; i < 7; i++)
            va.elem[i] = a[i];
        va.length = 7;
    }
    else {
        printf("Error");
        exit(ERROR);
    }
    // before insertion
    PrintList_Sq(&va);
    // after insertion
    if (InsertElem_Sq(&va, 4) != ERROR)
        PrintList_Sq(&va);
    else printf("Insert failed");
    if (InsertElem_Sq(&va, 9) != ERROR)
        PrintList_Sq(&va);
    else printf("Insert failed");
    if (InsertElem_Sq(&va, 0) != ERROR)
        PrintList_Sq(&va);
    else printf("Insert failed");
    if (InsertElem_Sq(&va, -5) != ERROR)
        PrintList_Sq(&va);
    else printf("Insert failed");

    DestroyList_Sq(&va);

    return 0;
}
```

向有序表 `va = {1, 2, 3, 5, 6, 7, 8}` 中依次插入元素 4、9、0 和 −5（检查在中间和两边插入）：


F:\2017\春季学期\数据结构\HW\hw1\2-11.exe

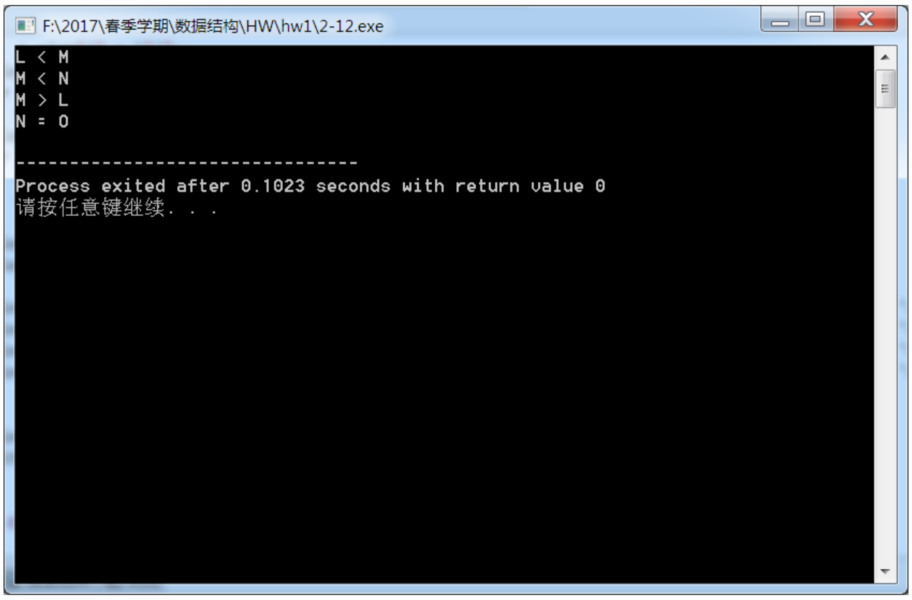
```
1 2 3 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
-5 0 1 2 3 4 5 6 7 8 9

--------------------------------
Process exited after 0.08039 seconds with return value 3221226356
请按任意键继续. . .
```

2.12 代码如下.

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

#define OK 1
#define ERROR 0
```

```c
#define OVERFLOW -2

#define GREATER 1
#define EQUAL 0
#define LESS -1

#define LIST_INIT_SIZE 100
#define LISTINCREMENT 10

// assume the elements are of type int
typedef int ElemType;
typedef int Status;

typedef struct {
    ElemType *elem;
    int length;
    int listsize;
} SqList;

int precede(ElemType x, ElemType y) {
    return x < y;
}

Status InitList_Sq(SqList *L) {
    L -> elem = (ElemType*)malloc(LIST_INIT_SIZE * sizeof(ElemType));
    if (!L -> elem) exit(OVERFLOW);
    L -> length = 0;
    L -> listsize = LIST_INIT_SIZE;

    return OK;
}

// compare two sequential lists
Status CompareList_Sq(SqList *A, SqList *B) {
    int p = 0;

    while (A -> elem[p] == B -> elem[p]) {
        ++p;
        if (p == A -> length)
            return (p == B -> length)? EQUAL: LESS;
    }
    if (precede(A -> elem[p], B -> elem[p]))
        return LESS;
    return GREATER;
}

void DestroyList_Sq(SqList *L) {
    free(L);
}

int main() {
    SqList L, M, N, O;
    int l[] = {1, 2, 3, 4, 5, 6, 7};
    int m[] = {1, 2, 3, 5, 6, 7, 8};
    int n[] = {1, 2, 3, 5, 6, 7, 8, 9};

    if (InitList_Sq(&L) && InitList_Sq(&M) && InitList_Sq(&N) && InitList_Sq(&
        O)) {
        int i;
        Status result;

        for (i = 0; i < 7; i++)
            L.elem[i] = l[i];
        L.length = 7;
        for (i = 0; i < 7; i++)
            M.elem[i] = m[i];
        M.length = 7;
        for (i = 0; i < 8; i++) {
            N.elem[i] = n[i];
            O.elem[i] = n[i];
        }
        N.length = 8;
        O.length = 8;

        printf("L %c M\n", ((result = CompareList_Sq(&L, &M)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
        printf("M %c N\n", ((result = CompareList_Sq(&M, &N)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
        printf("M %c L\n", ((result = CompareList_Sq(&M, &L)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
        printf("N %c O\n", ((result = CompareList_Sq(&N, &O)) == GREATER)? '>'
            : (result == LESS)? '<': '=');
    }
    else {
        printf("Error");
        exit(ERROR);
    }

    return 0;
}
```

创建四个线性表 L = {1, 2, 3, 4, 5, 6, 7}, M = {1, 2, 3, 5, 6, 7, 8}, N = O = {1, 2, 3, 5, 6, 7, 8, 9}，比较结果如下：



```
L < M
M < N
M > L
N = O

--------------------------------
Process exited after 0.1023 seconds with return value 0
请按任意键继续. . .
```

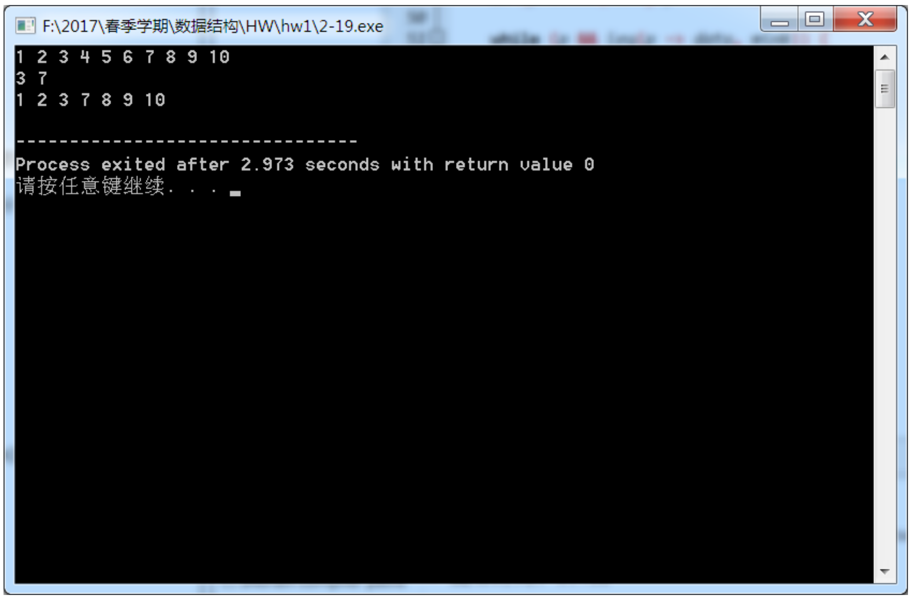
2.19 代码如下.

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

#define OK 1
#define ERROR 0

// assume the elements are of type int
typedef int ElemType;
typedef int Status;

typedef struct LNode {
    ElemType data;
    struct LNode *next;
} LNode, *LinkList;

int precede(ElemType x, ElemType y) {
    return x < y;
}

int leq(ElemType x, ElemType y) {
    return x <= y;
}

Status InitList_L(LinkList *L) {
    *L = (LinkList)malloc(sizeof(LNode));
    if (*L == NULL) return ERROR;
    (*L) -> next = NULL;

    return OK;
```

```c
}

Status CreateList_L(LinkList *L, int i) {
    LinkList p;

    p = (LinkList)malloc(sizeof(LNode));
    if (!p) return ERROR;
    p -> data = i;
    p -> next = (*L) -> next;
    (*L) -> next = p;

    return OK;
}

// delete all elements greater than mink and less than maxk
Status ListDeletebyBounds_L(LinkList *L, int mink, int maxk) {
    LinkList p = (*L) -> next, left, t;

    if (mink > maxk) return ERROR;
    if (precede(maxk, p -> data)) return OK;

    while (p && leq(p -> data, mink)) {
        left = p;
        p = p -> next;
    }
    if (p == NULL) return OK;

    while (p && precede(p -> data, maxk)) {
        t = p -> next;
        free(p);
        p = t;
    }
    left -> next = p;

    return OK;
}

Status PrintList_L(LinkList *L) {
    LinkList p = *L;

    while (p -> next != NULL) {
        p = p -> next;
        printf("%d ", p -> data);
    }
    printf("\n");

    return OK;
```

```c
}

void DestroyList_L(LinkList *L) {
    free(*L);
}

int main() {
    LinkList L;
    int i;

    InitList_L(&L);
    for (i = 10; i > 0; i--) {
        if (CreateList_L(&L, i) != ERROR) continue;
        else {
            printf("Error");
            exit(ERROR);
        }
    }
    PrintList_L(&L);

    int mink, maxk;
    scanf("%d%d", &mink, &maxk);
    if (ListDeletebyBounds_L(&L, mink, maxk) != ERROR)
        PrintList_L(&L);
    else {
        printf("Error");
        exit(ERROR);
    }
    DestroyList_L(&L);

    return 0;
}
```

创建单链表 L = {1, 2, 3, 5, 6, 7, 8, 9, 10}，输入 mink=$\boxed{3}$，maxk=$\boxed{7}$：



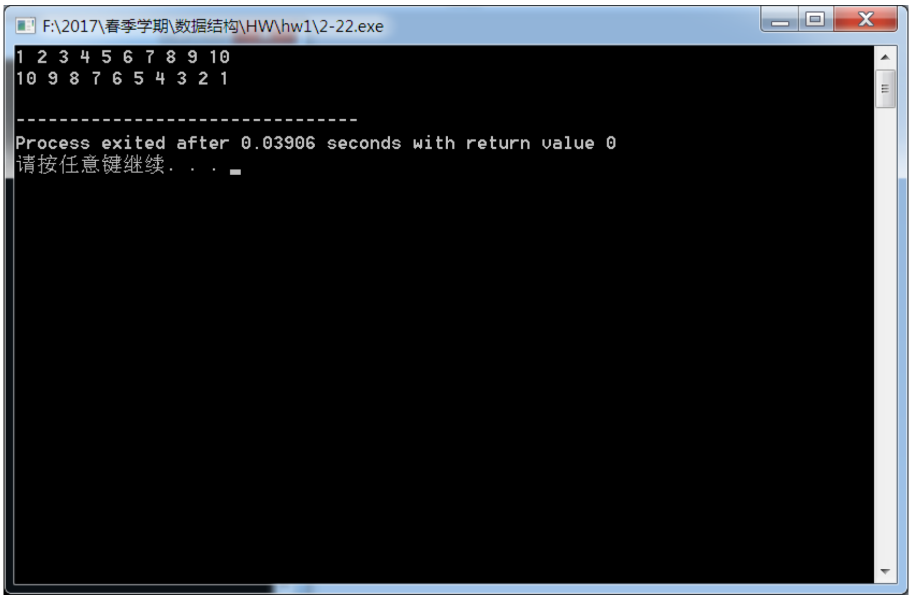
```
1 2 3 4 5 6 7 8 9 10
3 7
1 2 3 7 8 9 10

--------------------------------
Process exited after 2.973 seconds with return value 0
请按任意键继续. . .
```

时间复杂度为 $O(\texttt{maxk})$.

2.22 代码如下.

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

#define OK 1
#define ERROR 0

// assume the elements are of type int
typedef int ElemType;
typedef int Status;

typedef struct LNode {
    ElemType data;
    struct LNode *next;
} LNode, *LinkList;

Status InitList_L(LinkList *L) {
    *L = (LinkList)malloc(sizeof(LNode));
    if (*L == NULL) return ERROR;
    (*L) -> next = NULL;

    return OK;
}

Status CreateList_L(LinkList *L, int i) {
    LinkList p;

    p = (LinkList)malloc(sizeof(LNode));
    if (!p) return ERROR;
    p -> data = i;
    p -> next = (*L) -> next;
    (*L) -> next = p;

    return OK;
}

Status PrintList_L(LinkList *L) {
    LinkList p = *L;

    while (p -> next != NULL) {
        p = p -> next;
        printf("%d ", p -> data);
    }
    printf("\n");

    return OK;
}

void DestroyList_L(LinkList *L) {
    free(*L);
}

// in-place reversal
Status ReverseList_L(LinkList *L) {
    LinkList p = (*L) -> next, q;

    (*L) -> next = NULL;
    while (p) {
        q = p;
        p = p -> next;
        q -> next = (*L) -> next;
        (*L) -> next = q;
    }

    return OK;
}

int main() {
    LinkList L;
    int i;

    InitList_L(&L);
    for (i = 10; i > 0; i--) {
        if (CreateList_L(&L, i) != ERROR) continue;
        else {
            printf("Error");
            exit(ERROR);
        }
    }
    PrintList_L(&L);

    if (ReverseList_L(&L) != ERROR)
        PrintList_L(&L);
    else {
        printf("Error");
        exit(ERROR);
    }
    DestroyList_L(&L);

    return 0;
}
```

原地逆置单链表 `L = {1, 2, 3, 5, 6, 7, 8, 9, 10}`:



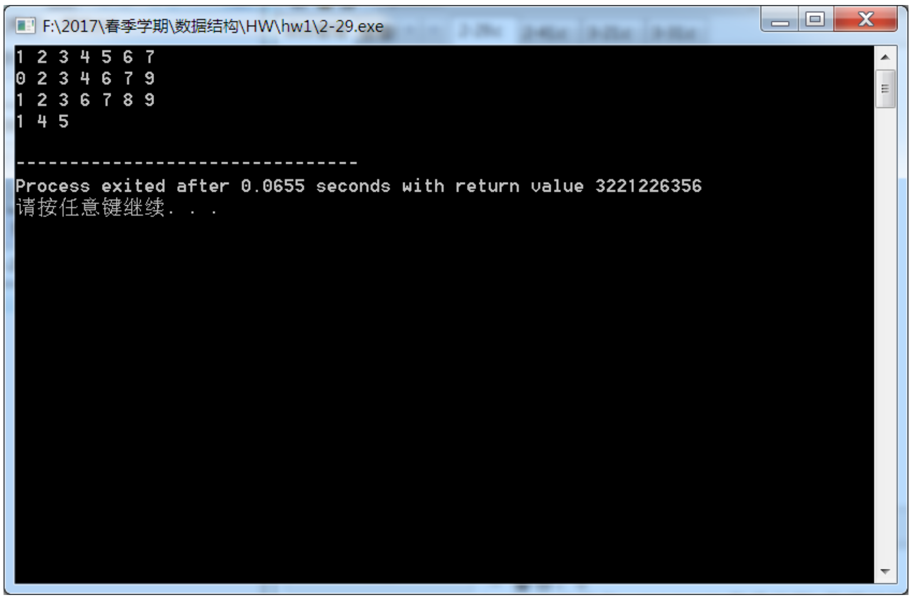
```
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1

--------------------------------
Process exited after 0.03906 seconds with return value 0
请按任意键继续. . .
```

2.29 代码如下.

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

#define OK 1
#define ERROR 0
#define OVERFLOW -2

#define FOUND 1
#define NONE 0

```

```
#define LIST_INIT_SIZE 100
#define LISTINCREMENT 10

// assume the elements are of type int
typedef int ElemType;
typedef int Status;

typedef struct {
    ElemType *elem;
    int length;
    int listsize;
} SqList;

int precede(ElemType x, ElemType y) {
    return x < y;
}

Status InitList_Sq(SqList *L) {
    L -> elem = (ElemType*)malloc(LIST_INIT_SIZE * sizeof(ElemType));
    if (!L -> elem) exit(OVERFLOW);
    L -> length = 0;
    L -> listsize = LIST_INIT_SIZE;

    return OK;
}

void PrintList_Sq(SqList *L) {
    int i;

    for(i = 0; i < L -> length; i++)
        printf("%d ", L -> elem[i]);
    printf("\n");
}

Status LocateElem_Sq(SqList *L, ElemType e) {
    int i;

    if (precede(e, L -> elem[0]) || precede(L -> elem[L -> length-1], e))
        return NONE;
    for (i = 0; i < L -> length && e != L -> elem[i]; i++)
        ;
    return (i == L -> length)? NONE: FOUND;
}

// note that the list is only non-decreasing so there may be multiple
    occurences
Status DeleteElem_Sq(SqList *L, ElemType e) {
    int i = 0, l, r;

    if (precede(e, L -> elem[0]) || precede(L -> elem[L -> length-1], e))
        return OK;
    for (i = 0; (i < L -> length) && (e != L -> elem[i]); i++)
        ;
    if (i == L -> length) return OK;
    l = i;
    while (e == (L -> elem[i]) && i < L -> length) i++;
    r = i;
    L -> length = L -> length - r + l;
    for (i = l; i < L -> length; i++)
        L -> elem[i] = L -> elem[i+(r-l)];

    return OK;
}

Status RemoveMutual_Sq(SqList *A, SqList *B, SqList *C) {
    int i;

    for(i = 0; i < B -> length; i++)
        if (LocateElem_Sq(C, B -> elem[i]) == FOUND) DeleteElem_Sq(A, B ->
            elem[i]);
    return OK;
}

void DestroyList_Sq(SqList *L) {
    free(L);
}

int main() {
    SqList A, B, C;
    int a[] = {1, 2, 3, 4, 5, 6, 7, 7};
    int b[] = {0, 2, 3, 4, 6, 7, 9, 9};
    int c[] = {1, 2, 3, 6, 7, 8, 9, 9};

    if (InitList_Sq(&A) && InitList_Sq(&B) && InitList_Sq(&C)) {
        int i;

        for (i = 0; i < 7; i++) {
            A.elem[i] = a[i];
            B.elem[i] = b[i];
            C.elem[i] = c[i];
        }
        A.length = 7;
        B.length = 7;
        C.length = 7;
        PrintList_Sq(&A);
        PrintList_Sq(&B);
        PrintList_Sq(&C);
    }
    else {
        printf("Error");
        exit(ERROR);
    }

    RemoveMutual_Sq(&A, &B, &C);
    PrintList_Sq(&A);
    DestroyList_Sq(&A);
    DestroyList_Sq(&B);
    DestroyList_Sq(&C);

    return 0;
}
```

从表 A = {1, 2, 3, 4, 5, 6, 7, 7} 中删除表 B = {0, 2, 3, 4, 6, 7, 9, 9} 和 C = {1, 2, 3, 6, 7, 8, 9, 9} 的共同元素:



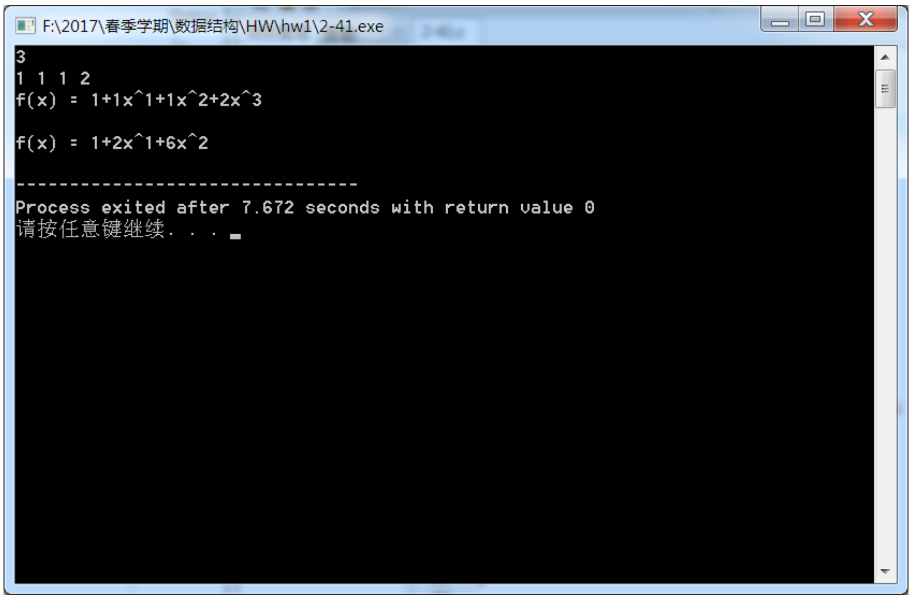
```
1 2 3 4 5 6 7
0 2 3 4 6 7 9
1 2 3 6 7 8 9
1 4 5

--------------------------------
Process exited after 0.0655 seconds with return value 3221226356
请按任意键继续. . .
```

2.41 代码如下.

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

#define OK 1
#define ERROR 0

typedef int Status;

```

```c
typedef struct {
    int coef;
    int exp;
} PolyTerm;

typedef struct PolyNode {
    PolyTerm data;
    struct PolyNode *next;
} PolyNode, *PolyLink;

typedef PolyLink LinkedPoly;

Status CreatePoly(LinkedPoly *L, int n) {
    *L = (LinkedPoly)malloc(sizeof(PolyNode));
    if (*L == NULL) return ERROR;
    (*L) -> next = NULL;

    int i;
    LinkedPoly p, prev, q;

    for (i = 0; i <= n; i++) {
        p = (LinkedPoly)malloc(sizeof(PolyNode));
        if (!p) return ERROR;
        scanf("%d", &p -> data.coef);
        p -> data.exp = i;
        prev = *L;
        q = (*L) -> next;
        while (q && (q -> data.exp < p -> data.exp)) {
            prev = q;
            q = q -> next;
        }
        p -> next = q;
        prev -> next = p;
    }

    return OK;
}

Status PrintPoly(LinkedPoly *L) {
    LinkedPoly p;
    p = *L;
    printf("f(x) = ");
    while (p -> next) {
        if (p -> next -> data.coef)
            if (p -> next -> data.exp != 0)
                printf("%dx^%d%c", p -> next -> data.coef, p -> next -> data.
                    exp, (p -> next -> next)? '+': ' ');
            else
                printf("%d%c", p -> next -> data.coef, (p -> next -> next)? '+
                    ': ' ');
        p = p -> next;
    }
    printf("\n");

    return OK;
}

Status DifferentiatePoly(LinkedPoly *L) {
    LinkedPoly p, q, t;

    q = *L;
    p = (*L) -> next;
    while (p) {
        if (p -> data.exp == 0) {
            t = p;
            p = p -> next;
            q -> next = p;
            free(t);
        }
        else {
            p -> data.coef *= p -> data.exp;
            --p -> data.exp;
            q = p;
            p = p -> next;
        }
    }

    return OK;
}

int main() {
    LinkedPoly P;
    int n;

    scanf("%d", &n);
    if (CreatePoly(&P, n) != ERROR) {
        PrintPoly(&P);
        DifferentiatePoly(&P);
        printf("\n");
        PrintPoly(&P);
    }
    else {
        printf("ERROR");
        exit(ERROR);
    }

    return 0;
}
```

首先读入多项式的最高项次数 $n = \boxed{3}$，然后次数由低到高依次读入各项系数$\boxed{1,1,1,2}$，对多项式 $2x^3 + x^2 + x + 1$ 求导：



```
3
1 1 1 2
f(x) = 1+1x^1+1x^2+2x^3

f(x) = 1+2x^1+6x^2

--------------------------------
Process exited after 7.672 seconds with return value 0
请按任意键继续. . .
```