**要求代码和实验报告规范，在算法思想中：对实验涉及的数据结构进行有效设计和分析；对算法进行分析并给出时间、空间复杂度的结论；清晰表达实验思路、出现的问题及解决方法。**

一、调试成功程序及说明

1、

题目：编程实现书P12 ADT List 基本操作13个：

* 1. 用顺序存储结构实现； （2）用链式存储结构实现；

算法思想：

运行结果：

结果分析：

附源程序。  

2、

题目：设元素值为整型的线性表L，分别采用顺序结构和链式结构存储，编写函数，实现线性表的就地逆置（书P31 4）。

算法思想：

顺序结构：

int InverseList(SeqList L)

{

int i, t;

for(i=0; i<L.length/2; i++)

{

t = L.pData[i];

L.pData[i] = L.pData[L.length-1-i];

L.pData[L.length-1-i] = t;

}

return OK;

}

主要思想：头尾元素交换、第二个和倒数第二个交换、第三个和倒数第三个交换，以此类推……直至遍历到中间元素。

链式结构：

Status ReverseList(ListInfo &L)

{

LNode \*p, \*q;

p = L.head->next;

q = p->next;

L.head->next = NULL;

while(p != NULL)

{

p->next = L.head->next;

L.head->next = p;

p = q;

if(q != NULL)

q = q->next;

}

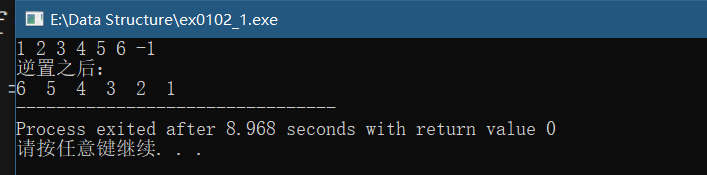
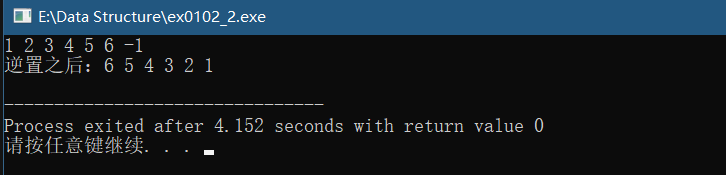
return OK;

}

主要思想：

将链表中的元素依次链接到表头的后面。

运行结果：



结果分析：

顺序结构：时间复杂度：O（n）

链式结构：时间复杂度：O（n）

附源程序。 

3、

题目：设线性表L，元素值为整型的且存在相同值，分别采用顺序结构和链式结构存储，编写函数，利用原空间，删除重复的元素值。

算法思想：

顺序结构：

void DeleteRepeat(SeqList &L)

{

int i, j;

for(i=0; i<L.length-1; i++)

for(j=i+1; j<L.length; j++)

if(L.pData[i] == L.pData[j])

DeleteElem(L, j);

}

主要思路：取一个元素，从这个元素的下一个元素开始遍历，若相等则删除。

链式结构：

Status DeleteRepeat(ListInfo &L)

{

int i, j=1;

LNode \*p = L.head->next;

LNode \*q;

while(p != NULL)

{

q = p;

while(q->next != NULL)

{

if(q->next->data == p->data)

{

LNode \*r = q->next;

q->next = r->next;

free(r);

}

else

q = q->next;

}

p = p->next;

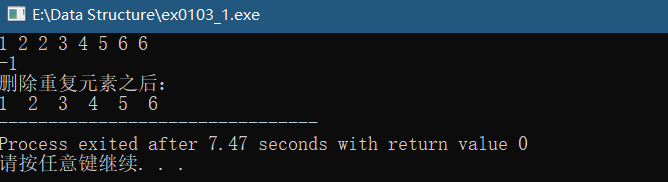
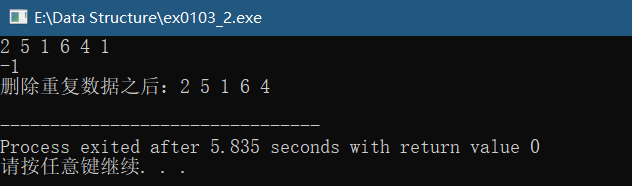
}

return OK;

}

主要思路：取一个元素，从这个元素的下一个元素开始遍历，若相等则删除。

运行结果：



结果分析：顺序结构：O（n^2）

链式结构：O（n^2）

附源程序。 

4、

题目：CSP 题目

问题描述：一次放学的时候，小明已经规划好了自己回家的路线，并且能够预测经过各个路段的时间。同时，小明通过学校里安装的“智慧光明”终端，看到了**出发时刻路上**经过的所有红绿灯的指示状态。请帮忙计算小明此次回家所需要的时间。

输入格式：

　　输入的第一行包含空格分隔的三个正整数 r、y、g，表示红绿灯的设置。这三个数均不超过 106。  
　　输入的第二行包含一个正整数 n，表示小明总共经过的道路段数和路过的红绿灯数目。  
　　接下来的 n 行，每行包含空格分隔的两个整数 k、t。k=0 表示经过了一段道路，将会耗时 t 秒，此处 t 不超过 106；k=1、2、3 时，分别表示**出发时刻**，此处的红绿灯状态是红灯、黄灯、绿灯，且倒计时显示牌上显示的数字是 t，此处 t 分别不会超过 r、y、g。

输出格式：

　　输出一个数字，表示此次小明放学回家所用的时间。

样例输入：

30 3 30  
8  
0 10  
1 5  
0 11  
2 2  
0 6  
0 3  
3 10  
0 3

样例输出：

46

样例说明：

　　小明先经过第一段路，用时10秒。第一盏红绿灯出发时是红灯，还剩5秒；小明到达路口时，这个红绿灯已经变为绿灯，不用等待直接通过。接下来经过第二段路，用时11秒。第二盏红绿灯出发时是黄灯，还剩两秒；小明到达路口时，这个红绿灯已经变为红灯，还剩11秒。接下来经过第三、第四段路，用时9秒。第三盏红绿灯出发时是绿灯，还剩10秒；小明到达路口时，这个红绿灯已经变为红灯，还剩两秒。接下来经过最后一段路，用时3秒。共计10+11+11+9+2+3=46秒。

评测用例规模与约定：

　　有些测试点具有特殊的性质：  
　　\* 前2个测试点中不存在任何信号灯。  
　　测试点的输入数据规模：  
　　\* 前6个测试点保证n≤103。  
　　\* 所有测试点保证n≤105。

算法思想：

Status Time(ListInfo L, int &time)

{

LNode \*p = L.head->next;

while(p != NULL)

{

int i, s;

s = red+yellow+green;

i = time/s;

if(p->k == 0)

time += p->t;

else if(p->k == 1)

{

if(p->t > time-i\*s )

time += (p->t - time+i\*s);

else if(time-i\*s>p->t && time-i\*s<p->t+green)

time += 0;

else if(time-i\*s>p->t+green && time-i\*s<p->t+green+yellow)

time += (yellow-(time-i\*s-p->t-green));

} else if(time-i\*s>p->t+green+yellow && time-i\*s<p->t+green+yellow+red)

time += (red-(time-i\*s-p->t-yellow-green));

else if(p->k == 2)

{

if(p->t > time-i\*s)

time += (p->t - time+i\*s);

else if( (time-i\*s > p->t) && (time-i\*s < p->t+red) )

time += (red-time+i\*s+p->t);

else if( (time-i\*s > p->t+red) && (time-i\*s < p->t+red+green) )

time += 0;

else if(time-i\*s>p->t+red+green && time-i\*s<p->t+red+green+yellow)

time += (yellow-(time-i\*s-p->t-red-green));

}

else if(p->k == 3)

{

if(p->t > time-i\*s)

time += 0;

else if(time-i\*s>p->t && time-i\*s<p->t+yellow)

time += (yellow-(time-i\*s-p->t));

else if(time-i\*s>p->t+yellow && time-i\*s<p->t+yellow+red)

time += (red-(time-i\*s-p->t-yellow));

else if(time-i\*s>p->t+yellow+red && time-i\*s<p->t+yellow+red+green)

time += 0;

}

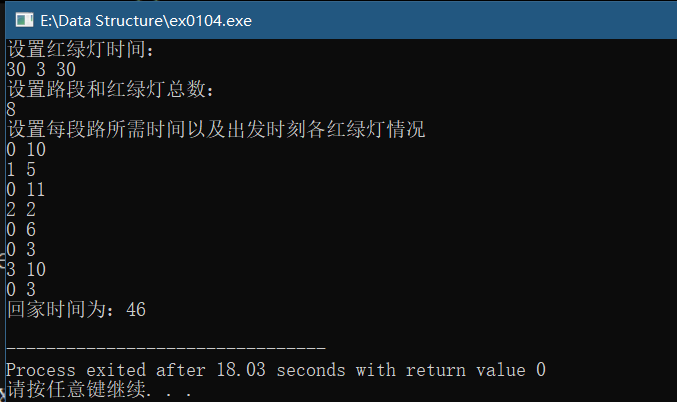
p = p->next;

}

return OK;

}

运行结果：



结果分析：

时间复杂度：O（1）

附源程序。

5、

题目：CSP 题目

问题描述：近来，跳一跳这款小游戏风靡全国，受到不少玩家的喜爱。  
　　简化后的跳一跳规则如下：玩家每次从当前方块跳到下一个方块，如果没有跳到下一个方块上则游戏结束。  
　　如果跳到了方块上，但没有跳到方块的中心则获得1分；跳到方块中心时，若上一次的得分为1分或这是本局游戏的第一次跳跃则此次得分为2分，否则此次得分比上一次得分多两分（即连续跳到方块中心时，总得分将+2，+4，+6，+8...）。  
　　现在给出一个人跳一跳的全过程，请你求出他本局游戏的得分（按照题目描述的规则）。

输入格式：

　　输入包含多个数字，用空格分隔，每个数字都是1，2，0之一，1表示此次跳跃跳到了方块上但是没有跳到中心，2表示此次跳跃跳到了方块上并且跳到了方块中心，0表示此次跳跃没有跳到方块上（此时游戏结束）。

输出格式：

　　输出一个整数，为本局游戏的得分（在本题的规则下）。

样例输入：

1 1 2 2 2 1 1 2 2 0

样例输出：

22

数据规模和约定：

　　对于所有评测用例，输入的数字不超过30个，保证0正好出现一次且为最后一个数字。

算法思想：

Status Score(ListInfo L, int &s)

{

LNode \*p = L.head->next;

int pre = 0;

while(p != NULL)

{

if(p->jump == 1)

{

s += 1;

pre = 0;

}

else if(p->jump == 2)

{

if(pre == 0)

{

s += 2;

pre += 2;

}

else if(pre%2 == 0)

{

s += (pre+=2);

}

}

p = p->next;

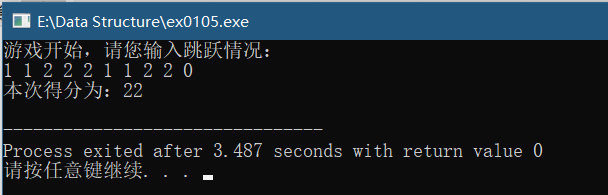
}

return OK;

}

主要思路：设置pre，若前一次没跳到中心，则归零，若跳到中心则+2。

运行结果：



结果分析：O（1）

附源程序。

6、

题目：

**1.19 试编写算法，计算的值并存入数组a[0..arrsize-1]的第i-1个分量中(i=1,2,…,n)。假设计算机中允许的整数最大值为maxint，则当n>arrsize或对某个，使时，应按出错处理。注意选择你认为较好的出错处理方法。**

算法思想：

if(k > arrsize-1) exit(0);//出错处理1

for(i=0; i<=k; i++)

{

if(i == 0) a[i] = 1;

else

{

if(2 \* i \* a[i-1] > MAXINT) exit(0);//出错处理2

else a[i] = 2 \* i \* a[i-1];

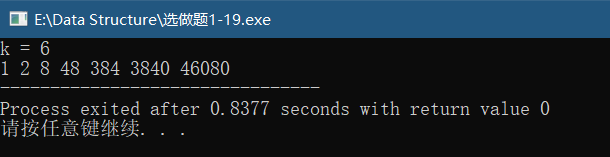
}

}

for(i=0; i<=k; i++)

printf("%d ", a[i]);

运行结果：



结果分析：从k = 7开始出错处理开始生效

附源程序。

7、

题目：

**1.20 试编写算法求一元多项式的值的值，并确定算法中每一语句的执行次数和整个算法的时间复杂度。注意选择你认为较好的输入和输出方法。本题的输入为，和，输出为。**

算法思想：

double polynomail(int a[], int i, double x, int n)

{

if(i > 0)

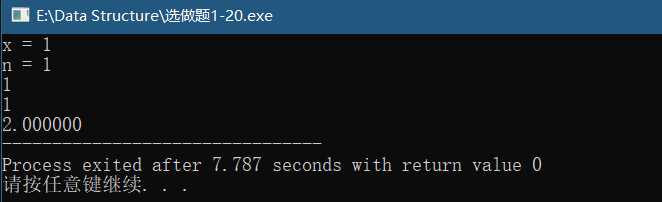
return (a[n-i] + polynomail(a,i-1,x,n) \* x);//递归

else

return a[n];

}

运行结果：



结果分析：

时间复杂度：O（n）

附源程序。

8、

题目：

**2.19 已知线性表中的元素以值递增有序排列，并以单链表作存储结构。试写一高效的算法，删除表中所有值大于mink且小于maxk的元素（若表中存在这样的元素），同时释放被删结点空间，并分析你的算法的时间复杂度（注意，mink和maxk是给定的两个参变量，它们的值可以和表中的元素相同，也可以不同）。**

算法思想：

Status DeleteBetween(ListInfo &L, int mink, int maxk)

{

LNode \*p, \*q, \*pre;

p = L.head->next;

pre = p;//p结点前驱

p = p->next;

while(p != NULL && p->data < maxk)

{

if(p->data <= mink)

{ //下一个

pre = p;

p = p->next;

}

else

{ //删除当前结点

pre->next = p->next;

q = p;

p = p->next;

free(q);

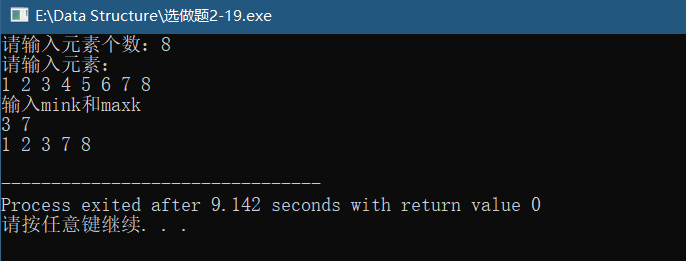
}

}

return OK;

}

运行结果：



结果分析：

时间复杂度：O（n）

附源程序。

二、未调试成功程序及说明

1、

题目：

算法思想：

错误原因：

附源程序。

2、

题目：

算法思想：

错误原因：

附源程序。

......

三、代码行数及小结