一、调试成功程序及说明

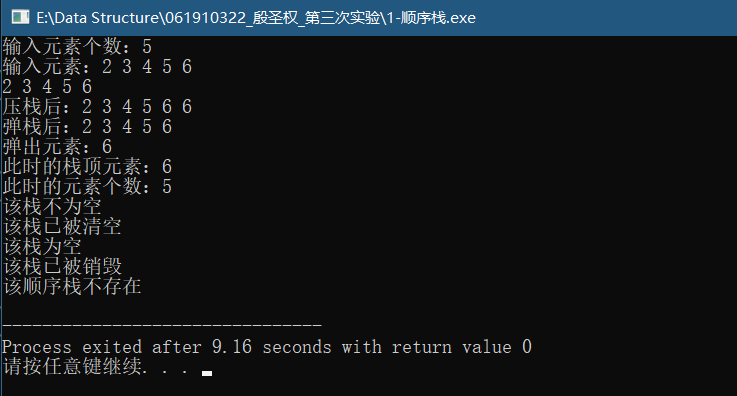
1、

题目：

编程实现书P32 ADT Stack 基本操作9个，用顺序存储结构实现；

算法思想：/

运行结果：



结果分析：/

附源程序。

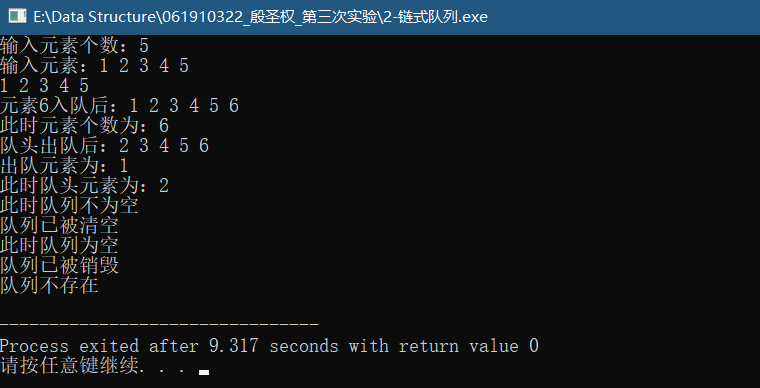
2、

题目：

编程实现书P48 ADT Queue 基本操作9个，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。

3、

题目：

利用栈操作实现八皇后问题求解。

算法思想：

//判断是否冲突

Status JudgeConfliction(SeqStack S, point newqueen)

{

point \*pCurqueen;

int x1, y1, x2, y2;

x1 = newqueen.col;

y1 = newqueen.row;

pCurqueen = S.pBase;

Status tag = OK;

while(pCurqueen < S.pTop-1)//与除栈顶之外的元素对比，判断是否冲突

{

x2 = pCurqueen->col;

y2 = pCurqueen->row;

if(x1 == x2)

{

tag = CONFLICT;

break;

}

if(abs(x1 - x2) == abs(y1 - y2))

{

tag = CONFLICT;

break;

}

pCurqueen++;

}

return tag;

}

//皇后放置-递归

Status PlaceQueen(int row, int &resultCount, SeqStack &StkQueen)

{

int col;

point curqueen;

for(col = 1; col <= N; col++)

{

curqueen.col = col;

curqueen.row = row;

Push(StkQueen, curqueen);//压栈

Status ret = JudgeConfliction(StkQueen, curqueen);//与除自己外的元素对比，判断是否冲突，若冲突则弹栈

if(ret == OK)//未冲突，轮到下一行

{

if(row < N)//未到达最后一行，继续递归下一行

PlaceQueen(row + 1, resultCount, StkQueen);

else

{

OutputResult(StkQueen, N);//最后一行的合适位置放置皇后之后输出结果

resultCount++;//结果数+1

}

}

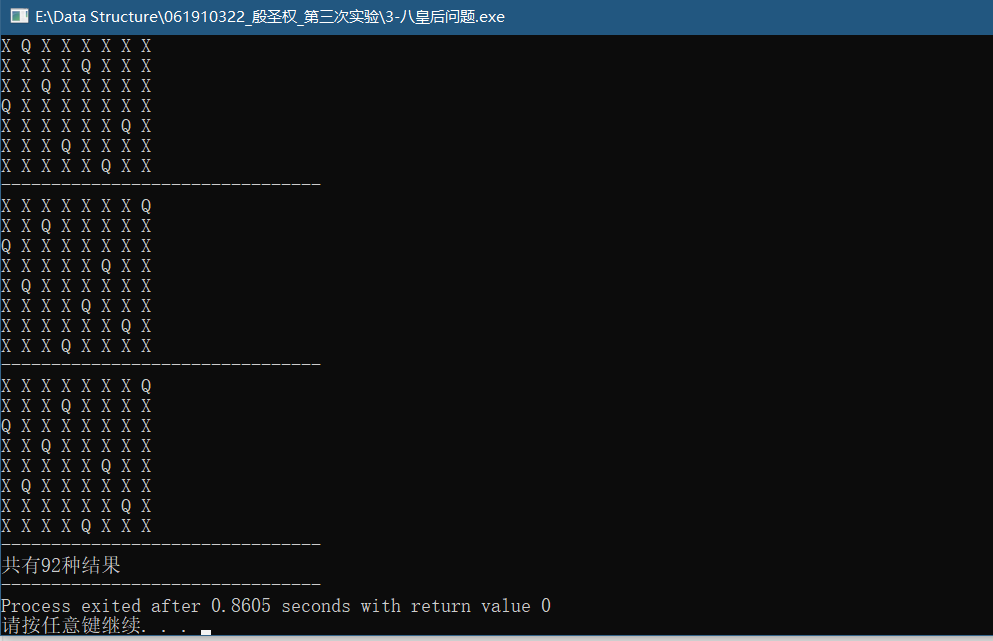
Pop(StkQueen, curqueen);//冲突，弹栈

}

return OK;

}

运行结果：



结果分析：时间复杂度：O（n）

附源程序。

4、

题目：

CSP题目

问题描述：小明今天生日，他有n块蛋糕要分给朋友们吃，这n块蛋糕（编号为1到n）的重量分别为a1, a2, …, an。小明想分给每个朋友至少重量为k的蛋糕。小明的朋友们已经排好队准备领蛋糕，对于每个朋友，小明总是先将自己手中编号最小的蛋糕分给他，当这个朋友所分得蛋糕的重量不到k时，再继续将剩下的蛋糕中编号最小的给他，直到小明的蛋糕分完或者这个朋友分到的蛋糕的总重量大于等于k。  
　　请问当小明的蛋糕分完时，总共有多少个朋友分到了蛋糕。

输入格式：

　　输入的第一行包含了两个整数n, k，意义如上所述。  
　　第二行包含n个正整数，依次表示a1, a2,…, an。

输出格式：

　　输出一个整数，表示有多少个朋友分到了蛋糕。

样例输入：

6 9  
2 6 5 6 3 5

样例输出：

3

样例说明：

　　第一个朋友分到了前3块蛋糕，第二个朋友分到了第4、5块蛋糕，第三个朋友分到了最后一块蛋糕。

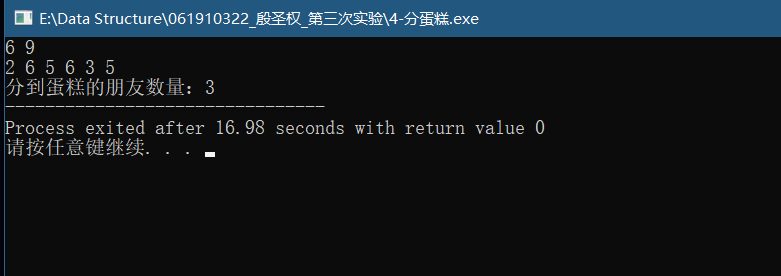
评测用例规模与约定：

　　对于所有评测用例，1≤n≤1000，1≤k≤10000，1≤ai≤1000。

算法思想：

依次弹栈，直至重量大于等于k，朋友数量+1。

运行结果：



结果分析：时间复杂度：O（n）

附源程序。

5、

题目：

CSP题目

问题描述：给定n个数，请找出其中相差（差的绝对值）最小的两个数，输出它们的差值的绝对值。

输入格式：

　　输入第一行包含一个整数n。  
　　第二行包含n个正整数，相邻整数之间使用一个空格分隔。

输出格式：

　　输出一个整数，表示答案。

样例输入：

5  
1 5 4 8 20

样例输出：

1

样例说明：

　　相差最小的两个数是5和4，它们之间的差值是1。

样例输入：

5  
9 3 6 1 3

样例输出：

0

样例说明：

　　有两个相同的数3，它们之间的差值是0.

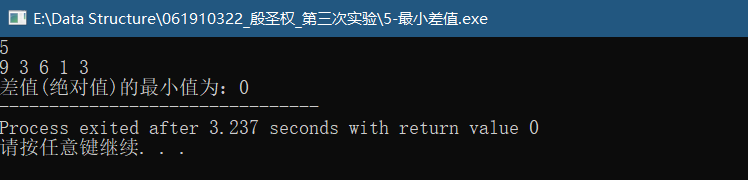
数据规模和约定：

　　对于所有评测用例，2≤n≤1000，每个给定的整数都是不超过10000的正整数。

算法思想：

先排序，依次进行相邻两数相减，与下一次的差比较，若更小，则更新最小值。

运行结果：



结果分析：时间复杂度：O（n）

附源程序。

6、

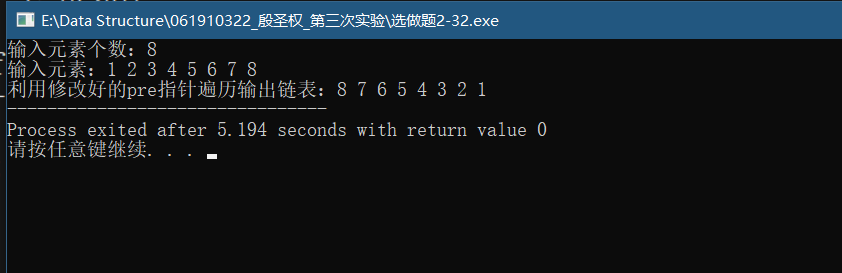
题目：

2.32 已知有一个单向循环链表，其每个结点中含三个域：pre，data和next，其中data为数据域，next为指向后继结点的指针域，pre也为指针域，但它的值为空，试编写算法将此单向循环链表改为双向循环链表，即使pre成为指向前驱结点的指针域。

算法思想：

遍历，将每一个结点的pre指针指向前驱。

运行结果：



结果分析：时间复杂度：O（n）

附源程序。

7、

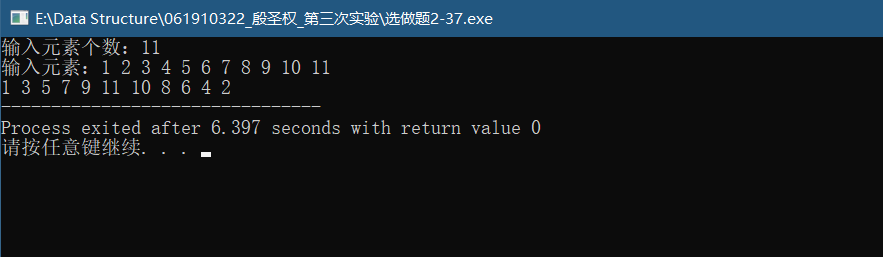
题目：

2.37 设以带头结点的双向循环链表表示的线性表。试写一时间复杂度O(n)的算法，将L改造为。

算法思想：

先断裂头尾，左端为头，然后每隔一个结点将结点摘除并依次接在最左端，直至最后一个结点为止。

运行结果：



结果分析：时间复杂度：O（n）

附源程序。

8、

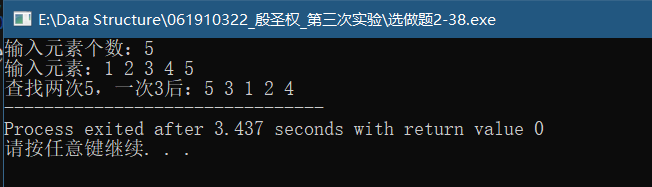
题目：

2.38 设有一个双向循环链表，每个结点中除有pre，data和next三个域外，还增设了一个访问频度域freq。在链表被起用之前，频度域freq的值均初始化为零，而每当对链表进行一次Locate(L,x)的操作后，被访问的结点（即元素值等于x的结点）中的频度域freq的值便增1，同时调整链表中结点之间的次序，使其按访问频度非递增的次序顺序排列，以便始终保持被频繁访问的结点总是靠近表头结点。试编写符合上述要求的Locate操作的算法。

算法思想：

先找到值为e的结点，freq+1， 然后对比前驱结点，若freq大则往前移，直至freq比前驱小。

运行结果：



结果分析：时间复杂度：O（n）

附源程序。

二、代码行数及小结

代码行数：

第一题：188

第二题：211

第三题：156

第四题：92

第五题：95

第六题：84

第七题：96

第八题：96

小结：

本次实验最大的收获就是能够运用递归思想解决问题，比如八皇后问题，利用递归思想能够比较简便地解决。