7.2

AStack<int> Input;

Input.push(5);

Input.push(1);

Input.push(3);

Input.push(4);

Input.push(6);

Input.print();

AStack<int> a, b;

while (!Input.isEmpty())

a.push(Input.pop());

b.push(a.pop());

while (!a.isEmpty())

{

while ((!b.isEmpty())&&(b.topValue() > a.topValue()))

Input.push(b.pop());

b.push(a.pop());

while (!Input.isEmpty())

b.push(Input.pop());

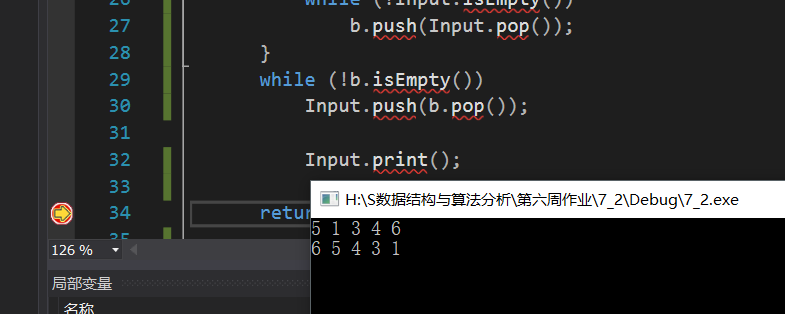
}

while (!b.isEmpty())

Input.push(b.pop());

Input.print();

return 0;



7.5

<a>

template <class Elem, class Comp>

void selsort(Elem A[], int n) { // Selection Sort

for (int i = 0; i<n - 1; i++) { // Select i’th record

int lowindex = i; // Remember its index

for (int j = n - 1; j>i; j--) // Find least value

if (Comp::lt(A[j], A[lowindex]))

lowindex = j; // Put it in place

if (lowindex != i) // Add check for exercise

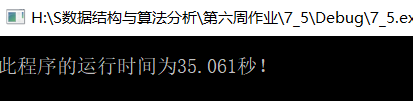
swap(A, i, lowindex);

}

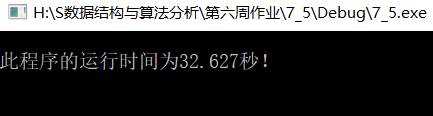
}

<b>应该不会,因为每次循环都要判断,在原来的基础上加了n次.

<c>加上检查



不加检查



7.6

插入排序稳定,只有值比上面记录的关键码小时才往上移动记录

冒泡排序稳定,只有低序号的关键码值比高序号的关键码值大,则交换顺序

选择排序不稳定,因为选择排序需要找到”第i小”的记录再放到数组的第i个位置,如果第i小的条件加入等于,那么排序将会避免移动相同的元素

Shell排序不稳定的,虽然每个子序列都是插入排序稳定,但是不同序列之间也有可能存在相同的元素,不保证一定保持原来顺序.

快速排序不稳定,因为同样分了序列.

归并排序稳定

堆排序同样因为有分成部分独立处理的做法,所以不稳定

分配排序稳定,

基数排序稳定