软件技术基础上机实践（一）：

排序与查找

一、上机实践

上机编程实现随机数的排序与查找，并调试通过，本次实践作业满分计 10 分。

作业说明：给定若干个随机数序列，分别用以下两种排序算法编程实现：

1. 简单插入排序
2. 快速排序

并在已排序的基础上用以下算法编程实现给定随机数的查找：

3. 二分查找

以上任务的主体程序由助教提供，同学们可把精力集中于算法本身的实现。

作业要求：

1. 必须提供完整的可编译并正确执行的源代码；
2. 应具有适当的注释、恰当的空格空行和缩紧等良好的代码风格；
3. 如果在完成以上任务的基础上，选择实现了其它排序算法，可酌情予以分数上的奖励。

二、参考伪代码

1． 简单插入排序算法

```
INSERTION-SORT (A, n)
1  ▷ Insert A[j] into the sorted sequence
2  for j ← 1 to n-1 do
3       key ← A[j]
4       i ← j – 1
5       while i >= 0 and A[i] > key do
6            A[i+1] ← A[i]
7            i ← i – 1
8       end
9       A[i+1] = key
10 end
```

2． 简单选择排序算法

```
SELECTION-SORT (A, n)
1  ▷ Exchange A[j] with the smallest
2  for j ← 0 to n-1 do
3       min ← j
4       i ← j + 1
5       while i < n do
6            if A[i] < A[min]
7                 min ← i
8            i ← i + 1
9       end
10      A[j] = A[min]
11 end
```

3． 冒泡排序算法

```
BUBBLE-SORT (A, n)
1  ▷ 将最小排序码交换到前面
2  for j ← 0 to n-2 do
3       tag ← 0
4        for i ← n-1 downto j+1 do
5           if A[i] < A[i-1] then
6                A[i] ↔ A[i – 1]
7                tag ← 1
8           end
9       end
10        if  tag = 0 break
11 end
```

4． 快速排序算法

主算法

```
QUICKSORT(A, p, r)
1 if p < r
2 then q ← PARTITION(A,p,r)
3    QUICKSORT(A,p,q-1)
4    QUICKSORT(A,q + 1,r)
5 end
```

分区算法

```
PARTITIONCLRS(A, p, r)
▷ 教材算法
1  pivot ← A[p]
2  i ← p,  j ← r
3  while i < j  do
4     while (i < j and A[j] >= pivot)
5        j ← j + 1
6     if (i < j) A[i] ↔ A[j]
7     while (i < j and A[i] <= pivot)
8        i ← i + 1
9     if (i < j) A[j] ↔ A[i]
10  end
11 A[i] ↔ A[p]
12 return i
```

```
PARTITIONCLRS(A, p, r)
▷ R. Sedgewick 算法
1  pivot ← A[p]
2  i ← p,  j ← r
3  while i < j  do
4     while (i < j and A[j] >= pivot)
5        j ← j + 1
6     while (i < j and A[i] <= pivot)
7        i ← i + 1
8     if (i < j) A[i] ↔ A[j]
9  end
10 A[i] ↔ A[p]
11 return i
```

```
PARTITIONCLRS(A, p, r)
▷ CLRS算法
1  pivot ← A[p]
2  i ← p
3  j ← p + 1
4  while j <= r
5     if A[j] < pivot then
6        i ← i + 1
7        A[i] ↔ A[j]
8     end
9     j ← j + 1
10 end
11 A[i] ↔ A[p]
12 return i
```