package com.my.algorithm.question;

import java.util.ArrayList;

import java.util.List;

public class AlgSortQuestion {

 // 插入排序

 public void insertSort(int[] list) {

  for (int i = 1; i < list.length; i++) {

   int temp = list[i];

   int j = i - 1;

   while (j >= 0 && list[j] > temp) {

    list[j + 1] = list[j];

    j--;

   }

   list[j + 1] = temp;

  }

  System.out.println("插入排序结果：");

  for (int i = 0; i < list.length; i++) {

   System.out.print(list[i] + " ");

  }

 }

 // 希尔排序:dk为1的时候就是插入排序

 public void shellInsert(int[] list, int start, int dk) {

  int i, j;

  for (i = start + dk; i < list.length; i = i + dk) {

   j = i - dk;

   int temp = list[i];

   while (j >= 0 && list[j] > temp) {

    list[j + dk] = list[j];

    j -= dk;

   }

   list[j + dk] = temp;

  }

 }

 // 希尔排序：将整个序列划分成若干子序列，分别进行直接插入排序。

 public void shellSort() {

  int[] a = { 2, 4, 3, 5, 6, 1, 7 };

  int i = 0, w = 2;

  while (i < w) {

   shellInsert(a, i, w);

   i++;

  }

  shellInsert(a, 0, 1);

  System.out.print("希尔排序结果：");

  for (i = 0; i < a.length; i++) {

   System.out.print(a[i]);

  }

 }

 // 冒泡排序

 public void bubbleSort(int[] list) {

  for (int i = 0; i < list.length; i++) {

   for (int j = 0; j < list.length - i - 1; j++) {

    if (list[j] > list[j + 1]) {

     int temp = list[j];

     list[j] = list[j + 1];

     list[j + 1] = temp;

    }

   }

  }

 }

 // 改进的冒泡排序

 public void bubbleSortImprove(int[] list) {

  int i = 0;

  boolean bool;

  do {

   bool = false;

   for (int j = 0; j < list.length - i - 1; j++) {

    if (list[j] > list[j + 1]) {

     int temp = list[j];

     list[j] = list[j + 1];

     list[j + 1] = temp;

     bool = true;

    }

   }

   i++;

  } while (bool == true && i < list.length);

 }

 // 快速排序：在快速排序中，记录的比较和交换是从两端向中间进行的，关键字较大

 // 的记录一次就能交换到后面单元，关键字较小的记录一次就能交换到前面单元，记录

 // 每次移动的距离较大，因而总的比较和移动次数较少。

 public void qucikSort(int[] list, int low, int high) {

  if (low < high) {

   int mid = partition(list, low, high);// 以a[mid]为基准元素将a[low:high]

   // 划分成3段a[low: mid -1],a[mid]和a[mid

   // +1:high]，使得a[low:

   // mid-1]中任何元素小于等于a[mid]，

   // a[mid +1:high]中任何元素大于等于a[mid]。下标mid在划分过程中确定。

   qucikSort(list, low, mid - 1);// 对左半段排序

   qucikSort(list, mid + 1, high);// 对右半段排序

  }

 }

 // 快速排序中获取中间值

 private int partition(int[] list, int low, int high) {

  int mid = list[low];// 用子表第一个元素作为轴

  while (low < high) {

   while (low < high && list[high] >= mid)

    high--;

   if (list[high] < mid)

    list[low] = list[high];

   while (low < high && list[low] <= mid)

    low++;

   if (list[low] > mid)

    list[high] = list[low];

  }

  list[low] = mid;

  return low;

 }

 // 字符串反转

 public String reverse(String str) {

  String str1 = "";

  for (int i = str.length(); i > 0; i--) {

   str1 += str.substring(i - 1, i);

  }

  return str1;

 }

 // 不开辟用于交换数据的临时空间，如何完成字符串的逆序(在技术一轮面试中，有些面试官会这样问)

 public void reverseNoSpace(String str) {

  char[] temp = str.toCharArray();

  for (int i = 0, j = str.length() - 1; i < j; i++, j--) {

   temp[i] ^= temp[j] ^= temp[i] ^= temp[j];

  }

  System.out.println(temp);

 }

 // 删除串中指定的字符（做此题时，千万不要开辟新空间，否则面试官可能认为你不适合做嵌入式开发）

 public void delChar(char[] str, char c) {

  int i, j = 0;

  for (i = 0; i < str.length; i++)

   if (str[i] != c)

    str[j++] = str[i];

  str[j] = '\0';

  System.out.println(str);

 }

 // 将一整数逆序后放入一数组中（要求递归实现）

 private List convertInt(int input, List<Integer> output) {

  if (input >= 10) {

   output.add(input % 10);

   return convertInt(input / 10, output);// input =

   // input/10;

   // 错误的赋值

  }

  if (input < 10)

   output.add(input % 10);

  return output;

 }

 // 递归实现回文判断（如：abcdedbca就是回文，判断一个面试者对递归理解的简单程序）

 public boolean isHuiWen(char[] str, int start, int length) {

  if (length - start <= 1)

   return true;

  else if (str[start] == str[length - 1]) {

   return isHuiWen(str, ++start, --length);

  } else {

   return false;

  }

 }

 // 组合问题（从M个不同字符中任取N个字符的所有组合）

 // public void combination(char[]source,char[]result,int

 // n){

 // if(n==1)

 // for(int i=0;i<source.length;i++)

 // System.out.print(source[i]);

 // else {

 // int i=source.length;

 // int j=result.length;

 // for(;i>=n;i--){

 // result[j]=source[i];

 // result[j+1]='\0';

 // combination(source, result, n-1);

 // }

 // }

 // }

 // 分解成质因数(如435234=251\*17\*17\*3\*2，据说是华为笔试题)

 public void prim(int m, int n) {

  if (m > n) {

   while (m % n != 0)

    n++;

   m /= n;

   prim(m, n);

   System.out.print(n + "\*");

  }

 }

 // 删除字符串中的数字并压缩字符串（神州数码以前笔试题），如字符串”abc123de4fg56”处理后变为”abcdefg”。

 // 注意空间和效率。（下面的算法只需要一次遍历，不需要开辟新空间，时间复杂度为O(N)）

 public void deleNum(String str) {

  int i, j;

  char[] strC = str.toCharArray();

  // //找到字符串中第一个数字的位置

  for (i = 0; (i < str.length() && strC[i] > '9' || strC[i] < '0'); i++)

   ;

  // 从串中第一个数字的位置开始，逐个放入后面的非数字字符

  for (j = i; j < str.length(); j++)

   if (strC[j] > '9' || strC[j] < '0')

    strC[i++] = strC[j];

  strC[i] = '\0';

  System.out.println(strC);

 }

 // 求两个串中的第一个最长子串（神州数码以前试题）。如"abractyeyt","dgdsaeactyey"的最大子串为"actyet"。

 public void maxSubStr(String str1, String str2) {

  int i = 0, j = 0, k = 0, index = 0, max = 0;

  char[] char1 = str1.toCharArray();

  char[] char2 = str2.toCharArray();

  for (i = 0; i < str1.length(); i++)

   for (j = 0; j < str2.length(); j++)

    for (k = 0; i + k < str1.length() && j + k < str2.length() && char1[i + k] == char2[j + k]; k++)

     if (k > max) {// 当出现大于前子串长度的子串，替换子串的位置和长度

      index = j;

      max = k;

     }

  System.out.println(str2.substring(index, index + max + 1));

 }

 // 寻找父串中子串首次出现的位置，实现strstr功能，即在父串中寻找子串首次出现的位置。

 // （笔试中常让面试者实现标准库中的一些函数）

 public void getPosition() {

  char[] str = "happybirthdy".toCharArray();

  char[] subStr = "birth".toCharArray();

  int i, j, k = 0;

  for (i = 0; i < str.length; i++) {

   for (j = 0, k = 0; j < subStr.length && str[i + j] == subStr[j]; j++) {

    k++;

   }

   if (k == subStr.length) {

    System.out.println("position:" + i);

    break;

   }

  }

 }

 // 编码实现字符串转整型的函数（实现函数atoi的功能），据说是神州数码笔试题。如将字符串

 // ”+123”->123, ”-0123”->-123,

 // “123CS45”->123,“123.45CS”->123, “CS123.45”->0

 public void str2Int(char[] str) {

  //char[] str = "-123.4CS4546".toCharArray();

  int i = 0, sign = 1, value = 0;

  if (str != null && str[0] > '9' && str[0] < '0') {

   value = 0; // 如果第一个元素为字母，直接赋值零

  } else {

   if (str != null && str[0] == '-' || str[0] == '+') {

    // 判断是否存在符号位

    i = 1;

    sign = (str[0] == '-' ? -1 : 1);

   }

   for (; str[i] >= '0' && str[i] <= '9'; i++)

    value = value \* 10 + (str[i] - '0');

  }

  System.out.println(sign \* value);

 }

 // 最大连续递增数字串

 public void subNum(String str) {

  char[] strChar = str.toCharArray();

  int length = 0;

  int start = 0;

  int j = 0;

  for (int i = 0; i < str.length(); i++) {

   if (strChar[i] <= '9' && strChar[i] >= '0')

    for (j = i + 1; j < str.length() && strChar[j] <= '9' && strChar[j] >= '0'

      && strChar[j] == (strChar[j - 1] + 1); j++)

     ;

   if (j - i > length) {

    start = i;

    length = j - i;

   }

  }

  System.out.println(str.substring(start, length + start));

 }

 // 2005年11月金山笔试题。编码完成下面的处理函数。函数将字符串中的字符'\*'移到串的前部分，

 // 前面的非'\*'字符后移，但不能改变非'\*'字符的先后顺序，函数返回串中字符'\*'的数量。

 // 如原始串为：ab\*\*cd\*\*e\*12，处理后为\*\*\*\*\*abcde12，函数并返回值为5。（要求使用尽量少的时间和辅助空间）

 public void change(String str) {

  int i, j;

  char[] tempChar = str.toCharArray();

  for (i = j = str.length() - 1; j >= 0; j--) {

   if (tempChar[i] != '\*')

    i--;

   else if (tempChar[j] != '\*') {

    tempChar[i] = tempChar[j];

    tempChar[j] = '\*';

    i--;

   }

  }

  System.out.println(tempChar);

  System.out.print("the num of \*:" + (i + 1));

 }

 //测试函数

 public int test(){

  int i=1;

  try {

   System.out.print("try:"+i);

   return i;

  } finally {

   ++i;

   System.out.print("finally:"+i);

  }

 }

 public static void main(String[] args) {

  int[] list = { 2, 3, 1, 6, 8 };

  AlgSortQuestion test = new AlgSortQuestion();

  // test.insertSort(list);

  test.shellSort();

  // test.bubbleSortImprove(list);

  test.qucikSort(list, 0, list.length - 1);

  System.out.print("快速排序结果：");

  for (int i = 0; i < list.length; i++) {

   System.out.print(list[i] + " ");

  }

  //System.out.println(test.reverse("abcd"));// 字符串反转

  List<Integer> output = new ArrayList();// 逆序存储整数

  //test.convertInt(123, output);

  for (int i = 0; i < output.toArray().length; i++) {

   //System.out.print(output.toArray()[i] + " ");

  }

  // 回文

  // char[] huiWen = { 'a', 'b', 'c', 'b', 'a' };

  // boolean b = test.isHuiWen(huiWen, 0, 5);

  // System.out.println("is hui wen :" + b + " ");

  // 组合字符

  // int num=3;

  // char[] source ={'a','b','c','d'};

  // char[] result = new char[num+1] ;

  // result[num]='\0';

  // if(num>0&&source.length>0&&source.length>=num)

  // test.combination(source, result, num);

  // 分解质因数

  // test.prim(435233, 3);

  // 删除字符串中的数字

  // test.deleNum("ab12de");

  // 求最长子串

  // test.maxSubStr("fcdefg", "acdef");

  // 删除字符串中指定的字符

  // test.delChar(huiWen, 'a');

  // 求最大连续递增数字串

  // test.subNum("ab12d12345");

  // 把字符串中的\*调到最前

  // test.change("abd\*d\*\*e");

  //获取字符串中第一个子串位置

  //test.getPosition();

  //不开辟空间，反转字符串

  //test.reverseNoSpace("abcd");

  //字符串转整形

  //test.str2Int("d-123.456cs".toCharArray());

  //System.out.println(test.test());

 }

}