插入排序练习题

1.输入一个正整数，将其各位数字进行调换以后获得一个最大整数，返回该最大整数。

例如输入n=2563;则返回6532。

函数名称：int Fun(unsigned int n)

函数功能：将一个正整数各位数字进行调换以后获得一个最大整数，返回该最大整数。

输入参数： int n：正整数

输出参数：无

返回值：int：最大整数

2. 已知数组A的元素值按非递减有序排列，编写一个函数删除多余相同元素，并返回新的数组长度。

函数名称：int Fun(int A[], int n)

函数功能：删除数组A的多余相同元素，并返回新的数组长度。

输入参数： int A[]：数组A

int n：数组A的长度

输出参数：int A[]：数组A

返回值：int：新的数组长度。

3. 编写一个函数删除数组A中元素值在[x,y]区间的所有元素，并返回新的数组长度。

函数名称：int Fun(int A[], int n, int x, int y)

函数功能：删除数组A中元素值在[x,y]区间的所有元素，并返回新的数组长度。

输入参数： int A[]：数组A

int n：数组A的长度

int x, int y,：闭区间的左右边界

输出参数：int A[]：数组A

返回值：int：新的数组长度。

4. 将数组A中的元素分成左右两部分，然后整体交换两部分的元素的位置，称为旋转向量。例如A[6]={1,2,3,4,5,6}，假设左部分长度r=4，则进行旋转后，A[6]={5,6,1,2,3,4} 。

编写一个函数，实现向量旋转。

函数名称：void Fun(int A[], int n, int r)

函数功能：将数组A中的元素分成左右两部分，然后整体交换两部分的元素的位置。

输入参数：int A[]：需要被旋转的向量

int n：向量的长度

int r：向量左半段长度

输出参数：int A[]：旋转后的向量

返回值：无

5. 插入排序在对几乎已经排好序的数据操作时，效率高，几乎可以达到线性排序的效率，但进行插入操作时，每次只能将数据移动一位，难免出现大量重复移动，如果能将元素尽可能快的移动到它“该去”的地方，将大大减少重复移动的次数，提高效率。

希尔排序是把全部元素分组排序，将所有距离为步长gap的元素放在同一个组中，通过“跳跃式移动”的方法，能让元素每次移动一大步，即步长gap>1，大大提高了移动的效率。一趟排序下来，虽然同组的元素没有挨在一起，各组元素相互隔开，但是由于每一组都已经各自排好序了，所以整个序列还是“基本有序”的。

然后再取越来越小的步长进行排序，直到步长gap=1时，就是普通的插入排序，但是到了这步，整个序列是“基本有序”了，直接插入排序也能高效完成。

实现算法的基本代码如下，请将语句1到语句3补充完整。

void ShellSort(int vec[], int n) //希尔排序

{

int i, j, gap;

int temp;

for (gap=n/2; gap>0; gap/=2)

{

for(i=gap; i<n; i++)

{

temp= ; //语句1

for(j=i; (j>=gap) && (temp<vec[j-gap]); j-=gap)//跳跃插入，跳跃距离为gap

{

vec[j]= ; //语句2

}

vec[j]= ; //语句3

}

}

}

6. 小明设计了一种纸牌游戏，每张纸牌上都有一个100以内的数字，每个数字对应的纸牌只有一张。初始时所有的纸牌从上到下按数字大小顺序叠成一堆，如果从中抽走一张，小明就按抽走的纸牌位置分为两堆。对每堆纸牌，小明采用记忆最顶端牌的数字和这堆牌的张数记录。如剩余的牌为1,2,3,5,6,11，则牌分为3堆，记录为(1,3),(5,2),(11,1)。

现在有些牌采用如下规则要放回牌堆，有4种情况：

1.上靠，例如放回的牌数字为7，则放至第二堆牌的最下方，原记录变成(1,3),(5,3),(11,1)。

2.下靠，例如放回的牌数字为10，则放至第三堆牌的最上方，原记录变成(1,3),(5,2),(10,2)。

3.上下靠，例如放回的牌数字为4，则合并第1堆和第2堆，原记录变成(1,6),(10,2)。

3.上下不靠，例如放回的牌数字为9，则新增一堆，原记录变成(1,3),(5,2),(9,1),(11,1)。

给定已有记录（分别用数组元素a[i]和b[i]存储第i堆牌的顶端牌数字和张数），

根据上述描述，输出插入数字为num(num > 1)的纸牌后的新记录，并返回新纪录的数量（牌堆的数量）。

实现算法的基本代码如下，请将语句1到语句4补充完整。

int Fun(int a[], int b[], int n, int key)//保证插入的纸牌数字大于1

{

int i = 0;

while (a[i] > 0 && a[i] < key) //查找牌key放的位置，i指向下一堆牌

i++;

i -= 1; //i指向上一堆牌

if (a[i]+b[i] == key) //上靠

{

if (a[i+1] == key+1)//上下靠

{

b[i] += 1 + b[i+1];

for (int j=i+1; j<n-1; j++)//合并堆

{

a[j] = ; //语句1

b[j] = ; //语句2

}

n--;

}

else //仅上靠

{

b[i] += 1;

}

}

else if (a[i+1] == key+1)//仅下靠

{

a[i+1] = ; //语句3

b[i+1] += 1;

}

else //上下不靠

{

for (int j=n; j>i+1; j--)//腾出位置，添加新堆

{

a[j] = ; //语句4

b[j] = b[j-1];

}

a[i+1] = key;

b[i+1] = 1;

n++;

}

return n;

}