```cpp
//日期: 2018/ 时间:
#include <stdio.h>
#include <stdlib.h>

//#define p1
//#define p2
// p1递归实现  p2非递归实现
#define p3
//给出归并排序每一趟结束时的序列

const int maxn=100;
//将数组A的[L1,R1]/[L2,R2]合并为有序区间 (L2 = R1 + 1)
void merge(int A[],int L1,int R1,int L2,int R2){
    int i=L1,j=L2;      //i指向A[L1],j指向A[L2]
    int temp[maxn], index=0;        //temp临时存放合并后的数组, index为其下标
    while(i <= R1 && j<=R2){
        if(A[i] <= A[j]){
            temp[index++]=A[i++];
        }else{
            temp[index++]=A[j++];
        }
    }
    while(i<=R1) temp[index++]=A[i++];
    while(j<=R2) temp[index++]=A[j++];

    for(int i=0;i<index;i++)
        A[L1+i]=temp[i];
}

#ifdef p1
//将array数组当前区间[left,right]进行归并排序
void mergeSort(int A[],int left,int right){
    if(left < right){
        int mid = (left+right)/2;
        mergeSort(A,left,mid);
        mergeSort(A,mid+1,right);
        merge(A,left,mid,mid+1,right);
    }
}
#endif

#ifdef p2
//非递归排序: 每次分组组内元素个数上限都是2的幂次。
//设置一个步长step, 令其初值为2,然后将每step个元素作为一组, 将其内部进行排序
//(即将组内的左边的(step/2)个元素 与 右边(step/2)个元素 合并, 而若元素个数不超过 (step/2), 则不操作)
//再令step乘以2,重复上面的操作, 直到step/2超过元素个数n
//以下代码, 数组A[]下标从1开始
void mergeSort(int A[]){
    //step为组内元素个数, step/2为左子区间元素个数, 注意等号可以不取
    for(int step=2;step/2 <= n;step*=2){
        //每step个元素一组, 组内前step/2个元素和后step/2个元素进行合并
        for(int i=1,i<=n;i+=step){      //一个step分成一个小组
            //对每一组
            int mid=i + step/2 -1;      //左子区间元素个数为step/2
            if(mid + 1 <= n){           //右子区间存在元素则合并
```

```
56                  //左子区间为[i,mid],右子区间为[mid+1,min(i+step-1,n)]
57                  merge(A,i,mid,mid+1,min(i+step-1,n));
58              }
59          }
60      }
61  }
62  #endif
63
64  #ifdef p3
65  //如果要求给出每一次归并排序后的序列，那么完全可以使用sort函数取代merge函数
66  void mergeSort(int A[]){
67      //step为组内元素个数，step/2为左子区间元素个数，注意等号可以不取
68      for(int step = 2;step/2 <= n;step*=2){
69          //每step个元素一组，组内[i,min(i+step,n+1)]进行排序
70          for(int i=1;i<=n;i+=step){
71              sort(A+i,A+min(i+step,n+1));
72          }
73
74          //此处可以输出归并排序的某一趟结束后的序列
75      }
76  }
77
78  #endif
79
80  int main(){
81
82
83      return 0;
84  }
85
86
```