Merge 1

```
1  void merge(int a[],int m_a[],int f,int m,int l)
2  {
3      int l_lower=mid-1,i=f,j=m, merged_i=f;
4
5      while(i<=l_lower&&j<=l)
6          if(a[i]<a[j]){
7              m_a[merged_i]=a[i];
8              i++;
9              merged_i++;
10         }
11         else{
12             m_a[merged_i]=a[j];
13             j++;
14             merged_i++;
15         }
```

Merge 2

```
16        while(i<=l_lower){
17            m_a[merged_i]=a[i];
18            i++;
19            merged_i++;
20        }
21
22        while(j<=l){
23            m_a[merged_i]=a[j];
24            j++;
25            merged_i++;
26        }
27        for(i=f;i<=l;i++)
28          a[i]=m_a[i];
29    }
```

Merge sort

```
1  void merge_sort_r(int a[], int m_a[],int f,int l)
2  {
3      if(l-f<=0)
4          return;
5
6      int mid=(l+f)/2+1;
7
8      merge_sort_r(a, m_a,f,mid-1);
9      merge_sort_r(a, m_a,mid,l);
10
11     merge(a, m_a, f, mid, l);
12
13     return;
14 }
```