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           {
8              m_a[merged_i]=a[i];
9              i++;
10             merged_i++;
11          }
12       else
13          {
14             m_a[merged_i]=a[j];
15             j++;
16             merged_i++;
17          }
```

Merge 2

```
18      while(i<=l_lower)
19          {
20              m_a[merged_i]=a[i];
21              i++;
22              merged_i++;
23          }
24
25      while(j<=l)
26          {
27              m_a[merged_i]=a[j];
28              j++;
29              merged_i++;
30          }
31      for(i=f;i<=l;i++)
32          a[i]=m_a[i];
33  }
```

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 }
```