

Swap

```
1 void swap(int a[], int i, int j)
2 {
3     int temp=a[i];
4     a[i]=a[j];
5     a[j]=temp;
6 }
```

Bubble sort

```
1 void bubble(int a[], int n)
2 {
3     int i, unfinished=1;
4
5     while(unfinished){
6         unfinished=0;
7         for(i=0; i<n-1; i++){
8             if(a[i]>a[i+1]){
9                 unfinished=1;
10                swap(a, i, i+1);
11            }
12        }
13    }
```

Quicksort - median

```
1  int median(int a[], int i, int j, int k)
2  {
3
4      if(a[i]>a[j]&&a[i]>a[k])
5          return (a[j]>a[k]) ? j : k;
6      if(a[i]<a[j]&&a[i]<a[k])
7          return (a[j]>a[k]) ? k : j;
8
9      return i;
10 }
```

Quicksort - wrapper

```
1 void quick(int a[], int n)
2 {
3     quick_r(a, 0, n-1);
4 }
```

Quicksort - recursive function 1, termination

```
1 void quick_r(int a[],int first , int last)
2 {
3
4     if (last<=first )
5         return ;
6
7     if (last==first+1)
8         if (a[first]<a[last])
9             return ;
10        else {
11            swap(a , first , last );
12            return ;
13        }
```

Quicksort - recursive function 2, choose pivot

```
15     int i=first ,j=last -1;  
16     swap(a ,median(a ,first , first +1,last ) ,last );
```

Quicksort - recursive function 3, partition

```
17     while ( i < j ) {  
18         while ( a [ j ] >= a [ last ] && j > first )  
19             j --;  
20         while ( a [ i ] < a [ last ] )  
21             i ++;  
22         if ( i < j )  
23             swap ( a , i , j );  
24     }  
25  
26     swap ( a , last , i );  
27  
28     quick_r ( a , first , i - 1 );  
29     quick_r ( a , i + 1 , last );  
30  
31 }
```