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        {
 7           unfinished=0;
 8           for(i=0;i<n-1;i++)
 9              if(a[i]>a[i+1])
10                 {
11                    unfinished=1;
12                    swap(a,i,i+1);
13                 }
14        }
15  }
```

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          {
12             swap(a,first,last);
13             return;
14          }
```

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        {
19          while(a[j]>=a[last]&&j>first)
20            j--;
21          while(a[i]<a[last])
22            i++;
23          if(i<j)
24            swap(a,i,j);
25        }
26
27      swap(a,last,i);
28
29      quick_r(a,first,i-1);
30      quick_r(a,i+1,last);
31
32  }
```