```
object ImperativeQuickSort {



 def sort(a: Array[Int]) {

  def swap(i: Int, j: Int) {
   val t = a(i); a(i) = a(j); a(j) = t
  }

  def sort1(l: Int, r: Int) {
   val pivot = a((l + r) / 2)
   var i = l
   var j = r
   while (i <= j) {
    while (a(i) < pivot) i += 1
    while (a(j) > pivot) j -= 1
    if (i <= j) {
     swap(i, j)
     i += 1
     j -= 1
    }
   }
   if (l < j) sort1(l, j)
   if (j < r) sort1(i, r)
  }

  if (a.length > 0)
   sort1(0, a.length - 1)
 }
```

```
  def println(ar: Array[Int]) {
    def print1 = {
      def iter(i: Int): String =
        ar(i) + (if (i < ar.length-1) "," + iter(i+1) else "")
      if (ar.length == 0) "" else iter(0)
    }
    Console.println("[" + print1 + "]")
  }

  def main(args: Array[String]) {
    var ar = Array(6,5,2,1,8);
    println(ar)
    sort(ar)
    println(ar)
  }

}
```

////////////////////////////////For Internals better version/////////////////////////////////////

```
object ImperativeQuickSort {

  def sort(a: Array[Int]) {

    def swap(i: Int, j: Int) {
```

```
    val t = a(i); a(i) = a(j); a(j) = t
  }

  def sort1(l: Int, r: Int) {
    val pivot = a((l + r) / 2)
    var i = l
    var j = r
    while (i <= j) {
      while (a(i) < pivot) i += 1
      while (a(j) > pivot) j -= 1
      if (i <= j) {
        swap(i, j)
        i += 1
        j -= 1
      }
    }
    if (l < j) sort1(l, j)
    if (j < r) sort1(i, r)
  }

  if (a.length > 0)
    sort1(0, a.length - 1)
}



def main(args: Array[String]) {
  var ar = Array(6,5,2,1,8);
  val list = ar.toList
```

```
    println(list);
    sort(ar)
    val list1 = ar.toList
    println(list1);
  }

}
```