Quicksort in C

```
// To sort array a[] of size n: qsort(a,0,n-1)
void qsort(int a[], int lo, int hi) {
{
  int h, l, p, t;

  if (lo < hi) {
    l = lo;
    h = hi;
    p = a[hi];

    do {
      while ((l < h) && (a[l] <= p))
          l = l+1;
```

```
      while ((h > l) && (a[h] >= p))
          h = h-1;
      if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
      }
    } while (l < h);

    a[hi] = a[l];
    a[l] = p;

    qsort( a, lo, l-1 );
    qsort( a, l+1, hi );
  }
}
```

Quicksort in Haskell

```
qsort []     = []
qsort (x:xs) = let (a,b) = part xs in
   qsort a ++ x : qsort b
   where
     part [] = ([],[])
     part (y:ys) | y < x     = (y:a,b)
                 | otherwise = (a,y:b)
                     where (a,b) = part ys
```

Quicksort in Ruby

```
def qsort(list)
  return [] if list.size == 0
  x, *xs = *list
  less, more = xs.partition{|y| y < x}
  qsort(less) + [x] + qsort(more)
 end
```

Quicksort in Ocaml

```
let rec quicksort = function
    | (x::xs) ->
        (quicksort (List.filter (fun i -> i < x) xs))
        @ [x] @
        (quicksort (List.filter (fun i -> i >= x) xs))
    | [] -> []
      ;;
```