Bubble Sort

Bubble Sort- Bubbling heavy to the end

To sort an array *A*[0..*N*-1]:

```
for (int last = N -1; last >= 1; last --)
{
  // Move the largest entry in A[0…last] to A[last]
  for (int index = 0; index <= last-1; index++)
  {
       //swap adjacent elements if necessary
        if (A[index] > A[index+1])
        {
            int temp = A[index];
            A[index] = A[index+1];
            A[index + 1] = temp;
        }
  }
}
```

Bubble Sort – Bubbling light to the front

To sort an array *A*[0..*N*-1]:

```
int A[] = {1, 10, 20, 15, 2, 3, 5,};

    for (int first = 0; first < A.length-1; first++)
    {
       for (int index = A.length -2; index >=0 ; index--)
       {
          if (A[index + 1] < A[index])
          {
             int temp = A[index + 1];
             A[index +1] = A[index];
             A[index] = temp;
          }
       }
    }
```

Selection Sort

Selection Sort- the largest number at the end

```
for (last = N -1; last >= 1; last --)
{
    //Move the largest entry in A[0…last] to A[last]
    //Determine position of largest in A[0..last] and store in maxIndex
    int maxIndex = 0;
    for (int index = 1; index <= last; index++)
    {
        if (A[index] > A[maxIndex])
            maxIndex = index;
    }
   // maxIndex is position of largest in A[0..last]
   // swap A[maxIndex] with A[last]
   int temp = A[maxIndex];
   A[maxIndex] = A[last];
   A[last] = temp;
}
```

Selection Sort - How would you put smallest number first?

```
for (first= 0; first < A.length; first++)
{
    //Move the smallest entry in A[0…last] to A[first]
    //Determine position of smallest in A[0..last] and store in minIndex
    int minIndex = A.length-1;
    for (int index = a.length - 2; index >=0; index--)
    {
        if (A[index] < A[minIndex])
            minIndex = index;
    }
   // maxIndex is position of largest in A[0..last]
   // swap A[maxIndex] with A[last]
   int temp = A[minIndex];
   A[minIndex] = A[first];
   A[first] = temp;
}
```

Insertion Sort

Insertion Sort

```
//A[0..0] is sorted
for (index = 1; index <= N -1; index ++)
{
    // A[0..index-1] is sorted
    // insert A[index] at the right place in A[0..index]
    int unSortedValue = A[index];
    scan = index;
    while (scan > 0 && A[scan-1] > unSortedValue)
    {
       A[scan] = A[scan-1];
       scan --;
    }
   // Drop in the unsorted value
   A[scan] = unSortedValue;
   // Now A[0..index] is sorted
}
// Now A[0..N -1] is sorted, so entire array is sorted
```

Quicksort

Quicksort

To sort the entire array A[0..N -1], simply call doQuicksort and pass it 0 and N-1 for start and end:

```
void Quicksort(int A[ ])
{
    doQuicksort(A, 0, N – 1);
}
```

Quicksort

- This recursive procedure will repeatedly partition the sublists until A[start..end] is sorted:

```
void doQuicksort(int A[ ], int start, int end)
{
    if (start < end)
    {
        // partition A[start..end] and get the pivot point
        int pivotPoint = partition(A, start, end);
        // recursively do the first sublist
        doQuicksort(A, start, pivotPoint-1);
        // recursively do the second sublist
        doQuicksort(A, pivotPoint+1, end);
    }
}
```

How to Partition A[start..end]

```
int partition(int A[ ], int start, int end)
{
    int pivotValue = A[start];
    endOfLeftList = start;
    // At this point A[endOfLeftList] == pivotValue
    for (int scan = start + 1; scan <= end; scan ++)
    {
        if (A[scan] < pivotValue)
        {
            endOfLeftList ++;
            swap(A, endOfLeftList, scan);
            // At this point A[endOfLeftList] < pivotValue
        }
    }
   // Move the pivotValue between the left and right sublists
   swap(A, start, endOfLeftList);
   return endOfLeftList;
}
```