```
MERGE SORT (TOP-DOWN)

// Merges A[low..mid] with A[mid+1..high].
// Assumes those ranges are already sorted.
void merge(vector<int> &A, int low, int mid, int high)
{
  // Copy A[low..high] to auxiliary space first.
  vector<int> aux(high+1);  // allocate some extra space
  for (int k = low; k <= high; k++)
    aux[k] = A[k];

  // Merge back to A[low..high]
  int i = low;      // index into left-hand portion of array aux[low..mid]
  int j = mid + 1;  // index into right-hand portion of array aux[mid+1..high]
  for (int k = low; k <= high; k++) // k = index into merged section of A[low..high]
  {
    if (i > mid)      // left portion is empty
    {
      A[k] = aux[j];  // take from right portion
      j++;            // advance right index
    }
    else if (j > high)  // right portion is empty
    {
      A[k] = aux[i];    // take from left portion
      i++;              // advance left index
    }
    else if (aux[j] < aux[i]) // current right item < current left item
    {
      A[k] = aux[j];  // take from right portion
      j++;            // advance right index
    }
    else // current left item < current right item
    {
      A[k] = aux[i];    // take from left portion
      i++;              // advance left index
    }
  }
}

void mergesort(vector<int> &A, int low, int high)
{
  if (high <= low) return;
  int mid = low + (high - low)/2;   // same as (low+high)/2
  mergesort(A, low, mid);           // recursively sort the left half
  mergesort(A, mid+1, high);        // recursively sort the right half
  merge(A, low, mid, high);         // merge the two sorted halves back together
}

void mergesort(vector<int> &A)
{
  mergesort(A, 0, A.size()-1);
}




MERGE SORT (BOTTOM-UP)

(Uses same merge function from above)

// len represents the size of the sub-vectors we are currently merging:
//    by merging sub-vectors of size 1, then 2, then 4, then 8...
// the inner loop iterates over all the sub-vectors of the size specified by
//    the outer loop.

void mergesort(vector<int> &A)
{
  int n = A.size();
  for (int len = 1; len < n; len *= 2)
    for (int low = 0; low < n - len; low += 2*len)
      merge(A, low, low + len - 1, min(low + 2*len - 1, n-1));
}
```