アルゴリズムとデータ構造 2015 第５回　　年　　組　　番　氏名

1. 以下に示したのは，クイックソートのプログラム例である。クイックソートは全体を部分に分割しながら同じ処理を繰り返していくという，代表的な**分割統治**アルゴリズムである。
　クイックソートでは，まず配列の中の適当な要素を pivot（**軸**）に選び，pivot 以下の要素を左側，pivot 以上の要素を右側に寄せることで，配列を左右 2 つの部分配列に分割する。さらにそれぞれの部分配列も同様の処理で分割し，要素が 1 つになるまで分割を繰り返すと，元の配列全体のソートが完了する。
　このプログラムでは，この分割を行うために，まず配列を左から走査して pivot 以上の要素を見つけ，次に右から走査して pivot 以下の要素を見つけ，左側で見つけた要素と右側で見つけた要素同士を交換することを繰り返す。空欄を埋めてクイックソートの関数を完成させよ。

```
public static void quicksort(double[] a, int left, int right) {

    if (left >= right) return;

    double pivot = a[(left + right) / 2];  // この例では中央の要素を pivot にしている

    int l = left, r = right;
    while (true) {
        while (a[l] < pivot) {  // 左から走査して pivot 以上の要素で止まる

        }
        while (a[r] > pivot) {  // 右から走査して pivot 以下の要素で止まる

        }
        if (l >= r) break;

        double t;


        l++; r--;
    }

    quicksort(a,          , r);

    quicksort(a, r + 1,          );
}
```

2. 以下は，ともに小さい順に整列された配列 a と配列 b を合併（マージ）し，全体が小さい順に整列された配列 c を構成して返す関数である。空欄を埋めて関数を完成させよ。

```
public static double[] merge(double[] a, double[] b) {

    double[] c = new double[a.length + b.length];

    int i = 0, j = 0, k = 0;
    while (k < a.length + b.length) {

        if (i >= a.length) c[k++] =

        else if (j >= b.length) c[k++] =

        else if (a[i] <= b[j]) c[k++] =

        else c[k++] =

    }
    return c;
}
```

3. マージソートも**分割統治**による高速なソートであり，問題 2 のアルゴリズムに従って配列の中の隣り合う整列済みの部分をマージしていくことで配列全体を整列させる。マージソートには，配列の 2 分割を要素数 1 個になるまで繰り返してから合併して戻していくトップダウンの方式と，隣り合う要素のマージからはじめて，隣り合う部分配列を結合して成長させていくボトムアップの方式がある。
　以下は，トップダウン方式の例である。配列を前半 a[left]～a[mid]と後半 a[mid+1]～a[right]に分割し，それぞれを再帰的にソートしてからマージする。この例の動作を理解し，空欄を埋めて関数を完成させよ。

```
public static void mergeSort(double[] a, int left, int right) {

    if (left >= right) return;

    int mid = (left + right) / 2;

    mergeSort(a,          , mid);    // 配列の前半部分をソート

    mergeSort(a,          , right);  // 配列の後半部分をソート

    // 配列の前半部分を退避する
    double[] b = new double[mid - left + 1];
    for (int h = 0; h < b.length; h++)
        b[h] = a[left + h];

    // 退避した前半部分と後半部分をマージする
    int i = 0, j = mid + 1, k = left;
    while (k <= right) {

        if (i >= b.length) a[k++] =

        else if (j > right) a[k++] =

        else if (b[i] <= a[j]) a[k++] =

        else a[k++] =

    }
    b = null;  // 退避用の配列を削除
}
```

4. クイックソートとマージソートの計算量を考える。簡単のため配列の要素数は $n=2^k$ とする。
(a) クイックソートで理想の pivot が選ばれた場合，配列はちょうど中央で 2 分割される。また，トップダウン方式のマージソートでは，配列はいつも中央で 2 分割される。このような場合，k 段階の分割で各部分配列の要素数はほぼ $n/2^k$ となる。要素数が 1 になるまでの分割の段階数を求めよ。

(b) 次に各段階での計算量を考える。クイックソートでは，次の分割のために全要素を pivot と比較するので，計算量は n に比例する。マージソートでも，退避とマージのために全要素をコピーするので，計算量は n に比例する。これに段階数である(a)を掛ければ，クイックソートおよびマージソートの全計算量が概算できる。計算量を O 記法で表せ。

(c) クイックソートで最悪の pivot が選ばれ続けた場合，分割の段階数は何段階になるか。