実験計画

出力プログラム

- kGenProgで今までに出力した受け入れ可能性が高い修正結果を使用
- 今回の実験ではこれまでに自分が分類したものを使用するが、今後は自動的に分類したものを使用
- 時間があればdefects4jの新しいプロジェクトも追加する

テスト関連

- 修正箇所を読み、より良い形に修正してもらう
  diffを修正していただく
- 修正にかかった時間を記録
- 修正はしてもしなくても良い、また修正不能があっても良い

出力プログラム(diff)

```
@@ -20,14 +20,9 @@
          Integer pivot = arr.get(0);
          ArrayList<Integer> lesser = new ArrayList<Integer>();
          ArrayList<Integer> greater = new ArrayList<Integer>();
+         lesser = quicksort(lesser);

-         for (Integer x : arr.subList(1, arr.size())) {
-             if (x < pivot) {
-                 lesser.add(x);
-             } else if (x > pivot) {
-                 greater.add(x);
-             }
-         }
+         for (Integer x:arr.subList(1,arr.size())){
                         + if (x < pivot){
                                 + lesser.add(x);
                         } else if (x >= pivot){
                                 greater.add(x);
                         }
                 }
          ArrayList<Integer> middle = new ArrayList<Integer>();
          middle.add(pivot);
          lesser = quicksort(lesser);
```

出力プログラム(.java)

```
package java_programs;

import java.util.*;

/*
 * To change this template, choose Tools | Templates * and open the template
in the editor. */
/**
 * * @author derricklin */public class QUICKSORT {
    public static ArrayList<Integer> quicksort(ArrayList<Integer> arr) {
        if (arr.isEmpty()) {
            return new ArrayList<Integer>();
        }

        Integer pivot = arr.get(0);
        ArrayList<Integer> lesser = new ArrayList<Integer>();
        ArrayList<Integer> greater = new ArrayList<Integer>();
      lesser = quicksort(lesser);

        for (Integer x:arr.subList(1,arr.size())){if (x < pivot)
{lesser.add(x);} else if (x >= pivot){greater.add(x);}}
        ArrayList<Integer> middle = new ArrayList<Integer>();
        middle.add(pivot);
        lesser = quicksort(lesser);
        greater = quicksort(greater);
        middle.addAll(greater);
        lesser.addAll(middle);
        return lesser;

    }

    public static ArrayList<Integer> flagment(ArrayList<Integer> arr) {
        if (arr.isEmpty()) {
            return new ArrayList<Integer>();
        }

        Integer pivot = arr.get(0);
        ArrayList<Integer> lesser = new ArrayList<Integer>();
        ArrayList<Integer> greater = new ArrayList<Integer>();

        for (Integer x : arr.subList(1, arr.size())) {
            if (x < pivot) {
                lesser.add(x);
            } else if (x >= pivot) {
```

```
                greater.add(x);
            }
        }
        ArrayList<Integer> middle = new ArrayList<Integer>();
        middle.add(pivot);
        lesser = quicksort(lesser);
        greater = quicksort(greater);
        middle.addAll(greater);
        lesser.addAll(middle);
        return lesser;

    }
}
```

経過時間：00分00秒

両方のプログラムを比較してもらい、出力プログラムをより読みやすい形に修正をしていただく。
例えば、正しくインデントをつけることや冗長な部分の削除や変更などを行ってもらいます。

計測内容

判定してもらうプログラムのリスト(判定用プログラム：出力プログラム)

バグ識別子	乱数シード										出力数
	0	1	2	3	4	5	6	7	8	9	
Math-46	5 : 100	6 : 100	0 : 100	0 : 100	0 : 100	1 : 100	2 : 100	1 : 100	0 : 100	1 : 100	16 : 1,000
Math-49	2 : 100	1 : 100	2 : 100	6 : 100	1 : 100	0 : 100	0 : 100	6 : 100	0 : 100	1 : 100	19 : 1,000
Math-72	5 : 100	2 : 12	1 : 100	2 : 43	5 : 71	1 : 9	4 : 17	2 : 17	2 : 36	0 : 19	24 : 424
Math-78	3 : 77	8 : 100	1 : 11	8 : 100	1 : 100	4 : 100	2 : 8	0 : 4	0 : 17	3 : 100	30 : 617
Math-80	4 : 100	1 : 100	11 : 100	3 : 100	0 : 100	1 : 100	1 : 100	1 : 100	3 : 100	1 : 59	26 : 959
Math-82	11 : 100	1 : 49	0 : 44	0 : 29	0 : 45	0 : 27	0 : 65	0 : 48	0 : 45	0 : 22	12 : 474
Math-85	0 : 100	1 : 100	0 : 100	1 : 100	0 : 100	2 : 100	1 : 100	0 : 100	0 : 100	0 : 100	5 : 1,000
合計	30 : 677	20 : 561	15 : 555	20 : 572	7 : 616	9 : 536	10 : 490	10 : 469	5 : 498	6 : 500	132 : 5,474