CS 1713 Exam 2 Solutions — Fall 2007

1) (10 points)

```
public boolean moreAsThanBs(String s) {
   int aCount = 0;
   int bCount = 0;
   for (int i=0;i<s.length();i++) {
      if (s.charAt(i) == 'A')
         aCount++;
      if (s.charAt(i) == 'B')
         bCount++;
   }
   return aCount > bCount;
}
```

2) (10 points)

```
public int largestRow(double[][] vals) {
   double maxVal = vals[0][0];
   int maxRow = 0;
   for (int i=0;i<vals.length;i++)
      for (int j=0;j<vals[i].length;j++)
         if (vals[i][j] > maxVal) {
             maxVal = vals[i][j];
             maxRow = i;
         }
   return maxRow;
}
```

3a) (5 points)

```
public int linearSearch(String[] array, String target) {
   for (int i=0;i<array.length;i++)
      if (array[i].equals(target))
         return i;
   return -1;
}
```

3b) (15 points)

```
public int binarySearch(String[] array, String target) {
       int low = 0;
       int high = array.length - 1;
       while (low <= high) {
          int mid = (low + high)/2;
          if (array[mid].equals(target))
             return mid;
          else if (array[mid].compareTo(target) > 0)
             low = mid + 1;
          else
             high = mid - 1;
       }
       return -1;
    }
   }
```

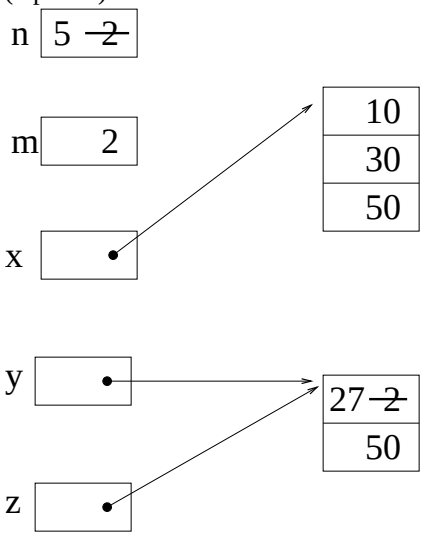
4) (5 points) a) 50 seconds b) 500 seconds c) about 8.8 seconds (1,000,000 elements requires about 20 times through the loop so 4 million requires about 22 times. 8*22/20 = 8.8.)

5) (20 points)

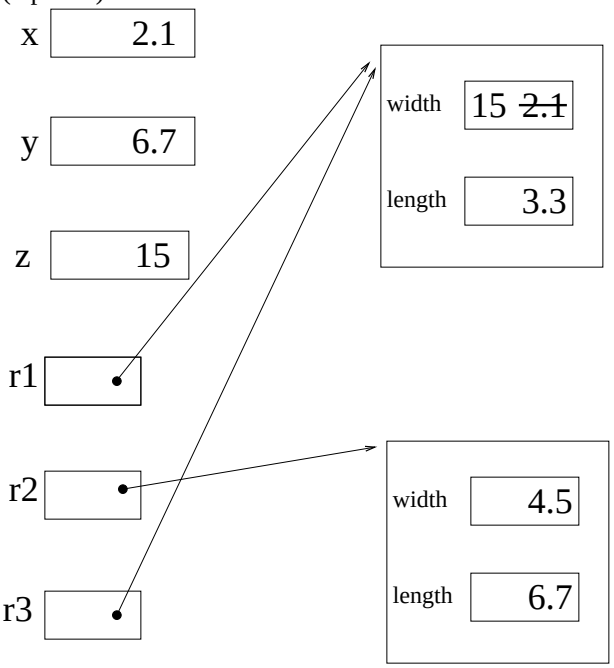
```
public static void selectionSort(double [] list) {
   for (int index = 0; index < list.length - 1; index++) {
       int min = findMinPosition(list, index);
       double temp = list[min];
       list[min] = list[index];
       list[index] = temp;
   }
}

public static int findMinPosition(double [] list, int start) {
   int min = start;
   for (int scan = start + 1; scan < list.length; scan++)
      if (list[min] > list[scan])
         min = scan;
   return min;
}
```

6) (7 points)



7) (8 points)



8) (20 points)

Selection Sort

0	1	2	3	4	5	6	7
56	26	73	35	19	17	11	43
11	26	73	35	19	17	56	43
11	17	73	35	19	26	56	43
11	17	19	35	73	26	56	43
11	17	19	26	73	35	56	43
11	17	19	26	35	73	56	43
11	17	19	26	35	43	56	73
11	17	19	26	35	43	56	73

Insertion Sort

0	1	2	3	4	5	6	7
56	26	73	35	19	17	11	43
26	56	73	35	19	17	11	43
26	56	73	35	19	17	11	43
26	35	56	73	19	17	11	43
19	26	35	56	73	17	11	43
17	19	26	35	56	73	11	43
11	17	19	26	35	56	73	43
11	17	19	26	35	43	56	73