COMP108 Algorithmic Foundations
Tutorial 4 (Suggested Solution and Feedback)

w/c 27th February 2017

1. The input numbers are stored in the array *data*[ ] and the number of integers is stored in the variable *count*.

```
// print the minimum number
public void findMin() {
    int loc = 0, i;
    for (i=1; i<count; i++) {
        if (data[i] < data[loc]) {
            loc = i;
        }
    }
    System.out.println();
    System.out.println("The minimum number is " + data[loc] + ".");
}
```

The try-catch structure is to handle exception. If the user input non-integer characters, an exception is thrown. The execution is then passed to the catch block without terminating the program.

The expression (condition ? $value_1$: $value_2$) will return $value_1$ if the condition is true and return $value_2$ if the condition is false.

```
// find if a certain number is in the array using SEQUENTIAL SEARCH
public void sfind() {
    int key=0;
    boolean found=false;
    try {
        System.out.print("Input number you want to find: ");
        key = keyboardInput.nextInt();
        for (int i=0; i<count && !found; i++) {
            if (data[i] == key) {
                found = true;
            }
        }
        System.out.println("The number " + key + " is" +
          (found ? "" : " NOT") + " found.");
    }
    catch (Exception e) {
        keyboardInput.next();
    }
}
```

The input numbers have been copied to the array $data2[\,]$ to be sorted. So you should do binary search on the array $data2[\,]$. The variable $first$ and $last$ point to the index of the first and last number in the current search range. The variable mid is calculated by $\frac{first+last}{2}$. In Java, if the variables are integers, then division will automatically truncate the decimal number and only return the integral part of the result. Sometimes this is handy (like our case), but sometimes this may create trouble if you indeed want the decimal result.

```
// find if a certain number is in the array using BINARY SEARCH
public void bfind() {
    int key=0, first, last, mid;
    boolean found=false;
    int[] data2 = new int[MaxCount];
    try {
        System.out.print("Input number you want to find: ");
        key = keyboardInput.nextInt();
        for (int i=0; i<count; i++) {
           data2[i] = data[i];
        }
        Arrays.sort(data2,0,count);
        first = 0;
        last = count-1;
        while (first <= last && !found) {
           mid = (first + last) / 2;
           if (data2[mid] == key) {
               found = true;
           } else {
               if (data2[mid] > key)
                  last = mid - 1;
               else
                  first = mid + 1;
           }
        }
        System.out.println("The number " + key + " is" +
          (found ? "" : " NOT") + " found.");
    }
    catch (Exception e) {
        keyboardInput.next();
    }
}
```