COMP108 Data Structures and Algorithms
Lab Exercises (Week 3)
(Suggested solution and Feedback)

w/c 22nd February 2021

1. Make sure you have the habit of entering your name and ID onto the beginning of programs.
2. **Task 1:** The method looks like this:

```
static void binary_search(int[] sortdata, int n, int key) {
    int count;
    boolean found;
    int first, last, mid;

    // start binary search on the sorted array called sortdata[]
    found = false;  // to indicate if the number is found
    first = 0;      // to indicate the beginning of the range being searched
    last = n-1;     // to indicate the end of the range being searched
    count = 0;      // to count how many comparisons are made

    while (first <= last && found==false) {
        mid = (first + last) / 2;
        if (sortdata[mid] == key) {
            found = true;
        } else {
            if (sortdata[mid] > key)
                last = mid - 1;
            else
                first = mid + 1;
        }
        count = count + 1;
    }
    System.out.print("The number " + key + " is ");
    if (found == false)
        System.out.print("not ");
    System.out.println("found by binary search and the number of comparisons used is
}
```

Feedback: The important point here is to set the correct while condition to continue the loop. When first is smaller than or equal to last, this means there is at least one more number we haven't considered yet.

When the two variables first and last are integers, then division by 2 will give us an integer (decimals discarded).

Comparing with the lecture notes, you have to translate the index range from 1..n to 0..(n-1).

3. **Task 2:**

- The method find_max looks like this:

```
// print and return the largest number in the array of size n
static int find_max(int[] data, int n) {
    int i, max;

    max = data[0];
    i = 1;
    while (i < n) {
        if (data[i] > max)
            max = data[i];
        i++;
    }
    System.out.println("find_max: The largest number is " + max + ".");
    return max;
}
```

Feedback: Some students has set `max = 0;` *before the loop. Note that this will not work if the numbers in the array are all negative. It's better to make the code to work for all cases and so better to set* `max = data[0];` *to start with.*

- The method find_secmax looks like this:

```
// print and return the second largest number in the array of size n
static int find_secmax(int[] data, int n) {
    int i, max, secmax;

    if (data[0] > data[1]) {
        max = data[0];
        secmax = data[1];
    } else {
        max = data[1];
        secmax=data[0];
    }

    i = 2;
    while (i < n) {
        if (data[i] > max) {
            secmax = max;
            max = data[i];
        } else if (data[i] > secmax)
            secmax = data[i];
        i++;
    }
    System.out.println("find_max: The largest number is " + max + ".");
    System.out.println("find_max: The second largest number is " + secmax + ".");
    return secmax;
}
```

Feedback: It is very important to set max and secmax correctly to start with. The idea is to set max to the larger of data[0] and data[1] while secmax to the smaller of data[0] and data[1]. Therefore, the following is wrong:

```
max = data[0];
secmax = data[1];
```

the following is wrong too:

```
max = data[0];
secmax = data[0];
```

- The method find_secmin is very similar to find_secmax.

```
// print and return the second smallest number in the array of size n
static int find_secmin(int[] data, int n) {
    int i, min, secmin;

    if (data[0] < data[1]) {
        min = data[0];
        secmin = data[1];
    } else {
        min = data[1];
        secmin=data[0];
    }

    i = 2;
    while (i < n) {
        if (data[i] < min) {
            secmin = min;
            min = data[i];
        } else if (data[i] < secmin)
            secmin = data[i];
        i++;
    }
    System.out.println("find_secmin: The smallest number is " + min);
    System.out.println("find_secmin: The second smallest number is " + secmin);
    return secmin;
}
```

Feedback: Similarly, we have to set min and secmin correctly to start with. The idea is to set min to the smaller of data[0] and data[1] while secmin to the larger of data[0] and data[1].