



## ASSIGNMENT 4 - SOLUTION

## GENERAL INFORMATION

This assignment is made public on Friday, February 28, 2002. The assignment is due on

Friday, March 7, 1 PM.

Hand in your assignment to the teaching assistant running your lab session.

The first page of your (written) assignment has to contain at least the following information:

- the course name (Introduction to Programming - Concepts and Tools)
- your name and your student number
- name and student number of the fellow student if you submit in pairs
- assignment number

*Please staple your assignment!*

You will get back the graded assignment one week after submission deadline.

## QUESTIONS

1. (*Arrays*)

What do the following three programs print on the screen? You should make a serious attempt to try to find out by inspecting the programs first without running them. (The programs are taken from the review section of chapter 5 of the book.)

For each of the question do *explain* why we have that particular output.

```
(a) public class Review1{  
  
    public static void main(String[] args){  
  
        int[] a1 = {1,2};  
        int[] a2;  
        a2 = a1;  
        a2[0] = 3;  
        System.out.println(a1[0]);  
    }  
  
}
```

```
(b) public class Review2{  
  
    public static void main(String[] args){  
  
        int[] a1 = new int[3];  
        int[] a2 = new int[10];  
        a1 = a2;  
  
        System.out.println(a1.length);  
    }  
  
}
```

```

(c) public class Review3{

    public static void main(String[] args){

        int i=2;
        int[] a = {1,2,3};
        System.out.println("goo = " + goo(i));
        hoo(a,1);
        hoo(a,i);
        for(i = 1; i < a.length; i++)
            System.out.println(i + ": " + a[i]);

    }

    static int goo(int i){
        return i+2;
    }

    static void hoo(int[] d, int n){
        for(int i = 0; i < d.length; i++)
            d[i] = goo(n);
    }

}

```

## SOLUTION

- (a) An array `a1` of length 2 is declared, a pointer to an array `a2` is declared. In the next line, the pointer `a2` is assigned the value of the pointer pointing to the array `a1`, i.e., both can be used to access the array defined first. Through `a2[0]` one cell of that array is changed to 3, and then that cell is printed by accessing it through `a1[0]`.

Output:

3

- (b) Two arrays of length 3 and 10 respectively are declared. The pointer `a1` is set to the value of `a2`, i.e., now points to the array `a2` having length 10.

Output:

10

- (c) The function `goo(i)` simply returns `i+2`, the function `hoo(arr, n)` stores `goo(n)`, i.e., `n+2` in each cell of the array `arr`.

The program initializes `i` to 2 and the array `a` is filled with values 1, 2, 3. Then the program types `goo(i)`, i.e., `goo(2)` on the screen, which is 4 (and the first line below).

Next `hoo(a,1)` is called, i.e., all cells of `a` are set to `1+2`, i.e., to 3.

Next `hoo(a,i)` is called, i.e., all cells of `a` are set to `i+2`, i.e., to 4.

Next the entry of the two cells of `a` are printed, showing thus the second and third line below.

Output:

goo = 4

1: 4

2: 4

## 2. (Character arrays, Palindrome)

- (a) Write a method of signature

```
boolean isPalindrome(char[])
```

that will take as input a character array and return `true` if that array is a palindrome<sup>1</sup>. Think first about the algorithm!

- (b) Write a program that will test your method. The program should be used the following way: If run using the command

```
java Palindrome "otto"
```

the user should get a message saying that he or she typed a palindrome.

Use the method

```
static char[] stringToCharArray(String s){
    return s.toCharArray();
}
```

to extract a character array from the input string.

### SOLUTION

- (a) 

```
static boolean isPalindrome(char[] arr){
    boolean flag = true;
    int i = 0;

    while(flag && i <= arr.length/2){
        if(arr[i]!=arr[arr.length-1-i])
            flag = false;
        i++;
    }
    return flag;
}
```

- (b) A possible solution is given below:

```
public class Palindrome{

    public static void main(String[] args){

        char[] input;

        input = stringToCharArray(args[0]);

        if(isPalindrome(input))
            System.out.println("You typed a palindrome!");
        else
            System.out.println("Sorry, what you typed is not a palindrome!");

    }

    // Put the code of isPalindrome here!

}
```

---

<sup>1</sup>Recall that a palindrome is a sequence of characters that reads the same way both forward and backward. Examples are `otto`, `121`, `i may yam i`.

### 3. (Random numbers, birthday problem)

In this question you are supposed to run a little simulation to find out the solution to the following problem: *How many people are needed in a room so that the probability that two of them have their birthday on the same day is greater or equal than 0.5.* If you haven't seen the answer you may be surprised.

To do so we will find, by a simulation, the probability that when  $n$  ( $1 \leq n \leq 100$ ) people are in a room two of them have their birthday on the same day. We will design the program using a top-down approach.

- (a) Design your main method that prints out for each number  $n$  between 1 and 100 the probability that among  $n$  people in a room at least two of them have their birthday on the same day. Call a method of signature

```
double simulation(int)
```

to do the bulk of the computation. That method will be designed further below.

- (b) Design the method `simulation`. You should make 10000 trials, and for each trial keep track of whether among the  $n$  people, two of them have their birthday on the same day. You should call a method of signature

```
boolean trial(int)
```

to be implemented later that will run 1 trial with  $n$  people, and return true or false depending on whether in that trial 2 or more people have their birthday on the same day.

- (c) Design the method `trial`.

- Use an array of integers of length 365 with one cell for each day of the year. For each day of the year the array will hold in cell  $i - 1$  the number of people that have their birthday on day  $i$ .
- For each of the  $n$  people generate a random number between 1 and 365 using the code `(int)(Math.random()*365 + 1)` to simulate their day of birth. Keep track of that birthday using the array.
- Once finished, search through the array to find out whether or not there is a day where 2 or more people have their birthday. Return that answer.

Submit your code, and one sample printout that shows the result of your program running.

## SOLUTION

```
(a) public static void main(String[] args){
    for(int i = 1; i <= 100; i++)
        System.out.println(i + ":\t" + simulation(i));
}

(b) static double simulation(int n){
    final int NUMBEROFTRIALS = 10000;
    int success = 0;

    for(int i = 1; i < NUMBEROFTRIALS; i++)
        if(trial(n))
            success++;

    return (double)success/NUMBEROFTRIALS;
}

(c) static boolean trial(int n){
    int[] days = new int[365];
    boolean flag = false;
    int j = 0;

    for(int i = 1; i <= n; i++){
        days[(int)(Math.random()*365)]++;
    }

    while(!flag && j<365){
        if(days[j]>=2)
            flag = true;
        j++;
    }

    return flag;
}
```

(d) `import tio.*;`

```
public class Birthday{  
    // put your three methods here!  
  
}
```

A possible output for the first 40 numbers (to save space for this solution) could be

1:	0.0
2:	0.0028
3:	0.0082
4:	0.0176
5:	0.0281
6:	0.042
7:	0.0528
8:	0.0785
9:	0.0924
10:	0.1152
11:	0.142
12:	0.158
13:	0.1916
14:	0.2218
15:	0.2558
16:	0.2825
17:	0.3153
18:	0.3492
19:	0.3851
20:	0.4039
21:	0.4337
22:	0.4853
23:	0.5118
24:	0.542
25:	0.5598
26:	0.603
27:	0.6288
28:	0.6531
29:	0.6776
30:	0.7036
31:	0.7333
32:	0.7466
33:	0.7733
34:	0.7998
35:	0.8167
36:	0.8317
37:	0.8499
38:	0.8579
39:	0.875
40:	0.8888

In particular, 23 people (only !) are enough to make a safe bet that 2 people in a room will have their birthday on the same day of the year.

#### 4. (Searching)

(Taken from Peter Sestofte's notes.)

- (a) Execute the binary search program from the lecture manually with the sorted array given below:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
17	20	28	33	35	39	39	51	52	57	62	69	74	84	84	86	88	95	97

with key being 84. Show the values of l, r, c and found before the loop, at pp3 in each iteration of the loop, and after the loop.

Use the following version of the binary search method which can also be downloaded from the course page:

```
static boolean binsearch(int[] arr, int key){
    /* Binary search for key in an ordered array arr.
       */

    boolean found = false;
    int l = 0;           // left marker
    int r = arr.length-1; // right marker
    int c;              // centre

    while(!found && l <= r){
        c = (l+r)/2;
        if(key < arr[c])
            { // search in lower part
              r = c-1;
            }
        else if(arr[c] < key)
            { // search in upper part
              l = c+1;
            }
        else
            found = true;

        // line pp3

    }

    return found;
}
```

- (b) Do the two programs for linear and binary search produce the right result when the array is empty (i.e. if the array is of length 0)? Give a reason for your answer.

## SOLUTION

- (a) The following program can be used to generate the correct answer

```
public class Searching{

    public static void main(String[] args){
        int[] data = {17, 20, 28, 33, 35, 39, 39, 51, 52,
                     57, 62, 69, 74, 84, 84, 86, 88, 95, 97};

        if(binsearch(data,84))
            System.out.println("Found!!");

    }

    static boolean binsearch(int[] arr, int key){
        /* Binary search for key in an ordered array arr.
        */

        boolean found = false;
        int l = 0;           // left marker
        int r = arr.length-1; // right marker
        int c = 0;           // centre

        System.out.println("l: " + l + "\tr: " + r + "\tc: " + c + "\tfound: " + found);
        while(!found && l <= r){
            c = (l+r)/2;
            if(key < arr[c])
                {// search in lower part
                    r = c-1;
                }
            else if(arr[c] < key)
                {// search in upper part
                    l = c+1;
                }
            else
                found = true;

            // line pp3
            System.out.println("l: " + l + "\tr: " + r + "\tc: " + c + "\tfound: " + found);
        }
        System.out.println("l: " + l + "\tr: " + r + "\tc: " + c + "\tfound: " + found);
        return found;
    }
}
```

As a result of running the program you will find

```
java Searching
l: 0   r: 18  c: 0   found: false
l: 10  r: 18  c: 9   found: false
l: 10  r: 18  c: 14  found: true
l: 10  r: 18  c: 14  found: true
Found!!
```

- (b) They do! For linear search note that the loop checks for `i<arr.length`, which is false, so the loop will not be entered, and we return immediately `found`, which is false. For binary search the same reasoning holds, this time since the left marker `l` is set to 0, whilst the right one `r` is set to `arr.length-1`, which is -1. The loop checks for `l<=r`, which is false, so the loop is not entered and `found` is returned.