Exercise Sheet 8

Issue Date: December 12th, 2023
Due Date: December 18th, 2023 – 10:00 a.m.
$\sum$ 10 Points

Konzepte der Informatik INF-11700
Winter 2023/2024

Universität Konstanz

University of Konstanz
Dr. Barbara Pampel
Sabrina Jaeger-Honz

Programming Paradigms & Recap

Exercise 1: Programming Paradigms (*2 points*)

Briefly describe (one short sentences) the basic characteristics of the following paradigms:

a) imperative
b) procedural
c) functional
d) logic

Hint: Look at our slides and do NOT copy from some online source! ;-)

Exercise 2: Reading Code (*2 points*)

The Pascal program in Listing 1 (on the next page for layout-reasons) picks a random number and compares it with another random number. After at most 10 comparisons the program should stop. Which problem does occur because of the GOTO-statements?
You can try out the code in an online-Compiler like
https://www.jdoodle.com/execute-pascal-online/

Exercise 3: QuickSort (*2 points*)

Given the Array A:

10	4	13	5	1	6	3

Sort A using quickSort. Visualize the progress just like on the lecture slides (06_2, slide 18) by printing (in a new line) at the beginning of each call of quickSort and after every change made on the array

- the array
- the positions of the pivot p and the pointers i and j

Listing 1 Random guess

```
PROGRAM EvilGOTO;
Var rand,count,guess : integer;
LABEL correct, loop, wrong, ende;

begin
  Randomize; {initialize to pick random numbers}
    count :=0;
    repeat
        loop:
        count := count+1;
        {Random(i) returns a random number between 0 and i}
        rand := Random(10);
        guess := Random(10);
        if rand = guess
        then
            goto correct
        else
            goto wrong
    until count > 10;
    goto ende;
    correct :
        WriteLn('correct');
        goto ende;
    wrong :
        WriteLn('false guess');
        goto loop;
    ende :
        WriteLn('ende');
end.
```

Exercise 4: InsertionSort – Analysis (*4 points*)

Give the run time complexity class ($\mathcal{O}(\ldots)$) InsertionSort belongs in:

a) (2 points) best-case

b) (2 points) worst-case

Give an example - input for each, which results in this run time.

Algorithm 1 InsertionSort

Input: array A containing comparable items
Result: non-descendingly sorted array A

```
1 for i = 2, ..., n do
2     j ← i
3     while j > 1 ∧ A[j] < A[j - 1] do
4         swap A[j] and A[j - 1]
5         j ← j − 1
```