



ASSIGNMENT 2

GENERAL INFORMATION

This assignment is made public on Friday, February 14, 2003. The assignment is due on

Friday, February 21, 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. (*Arithmetic expressions*)

For each of the following expressions give the correct bracketing showing the order in which the expression is evaluated, and give the result under the assumption that the variables are declared as

```
int a = 1, b = 2, c = 3, d = 4, e = 2147483647;
```

An example is the expression `b*c*d+a` which is evaluated as `((b*c)*d)+a` with result 25. (Note: In programming practice many of the expressions below should be avoided since they are hard to read. Instead, expressions should contain brackets making the evaluation clear.)

- (a) `a+b*c*d`
- (b) `a* ++b+c*d`
- (c) `-a++ +b/b`
- (d) `e+a-b`
- (e) `e+a`
- (f) `2+a/b`
- (g) `2.0+a/b`

2. (*Boolean expressions*)

What is the truth value of the following expressions when

```
boolean a = true, b = true, c = false, d = false;
```

- (a) `!true && true`
- (b) `!a||!b;`
- (c) `a&&b||c`
- (d) `a&&b==false;`
- (e) `a||!(b&&c);`
- (f) `!!(d||!c&&a)`

Suppose `x` and `y` are declared of type `int`, and `done` is a boolean variable. Write statements that do the following:

- (a) increment `x` by 1 if `done` is false;
- (b) decrement `x` by 1 if `x` is less or equal to `y`.

3. (*Code reading, loops*)

Consider the following program `Triangle.java` which can be downloaded from the course homepage:

```
/* Assignment 2
 *
 * Program written by Carsten Butz
 * September 3, 2002
 */

import tio.*;

public class Triangle {

    public static void main (String[] args){

        int input;
        int i,j;

        System.out.print("Please enter a number: ");
        input = Console.in.readInt();
        System.out.println();

        // Here start the loops that you are supposed to
        // rewrite:

        for(i=1; i<=input; i++)
        {
            for(j=1; j<=i; j++)
                System.out.print("*");

            System.out.print("\n");
        }

        // end of loops

        // next comes a separator
        System.out.println("\n-----\n");

        // insert your code here using while loops

        // next comes a separator
        System.out.println("\n-----\n");

        // insert your code here using do loops

        // next comes a separator
        System.out.println("\n-----\n");

    }
}
```

- (a) Explain in your own words what the program does (up to line `// end of loops`).

- (b) Explain the algorithm underlying the program.
- (c) At the two places marked `// insert your code here` insert code that will produce the same shape on the screen, but using `while` loops or `do` loops respectively. Submit a copy of your program.
Hint: Copy first the two `for` loops as they stand. Then replace the inner `for` loop by a `while` loop. Compile your program and make sure that it works properly before doing the next step. Replace the outer `for` loop by another `while` loop. Compile your program again and make sure that it still works properly. Once you are satisfied proceed similarly to rewrite the two `for` loops using `do` loops.

4. (*Prime numbers testing, loops*)

A *prime* number is a positive integer greater or equal to 1 that is only divisible by itself. Examples are 2, 3, 5, 7, 11, 13, 17 . . . , while 9 isn't (it is divisible by 3) and 15 isn't (because it is divisible by 3 and 5).

Write a program that asks the user to input an integer number and outputs whether or not the number is a prime number. An algorithm using a boolean flag might be as follows:

```
input a number n
isprime is true
while d greater than 2 and is less than or equal to square root of n
do the following:
    if d divides n then isprime is false
    increment d by 1

if n equals 1 or isprime is false print that n is not a prime
otherwise print that n is a prime.
```

- (a) Explain why it is enough to search for a divisor of `n` only up to square root of `n`.
- (b) Explain the role of the boolean flag `isprime`.
- (c) Argue that (i.e., give an explanation why) the algorithm above is correct, i.e.,
 - i. if `n` is a prime then at the end of the `while` loop the flag `isprime` is true;
 - ii. if `n` is composite (the product of two numbers) then at the end of the `while` loop the flag `isprime` is false.

Write your program and submit a paper copy of the code.

5. (*Prime number generation, iterated loops*)

You are supposed to write a program that asks the user to input a number and outputs all prime numbers smaller than or equal to that number. For this problem you can use some of the code you wrote for the previous or the previous but one question.

- (a) Write an algorithm that will solve the problem similar in style to the algorithm above that tested for a number being prime.
- (b) Write a program that will do the task.

Submit a paper copy of your program.