Computer Science and Mathematics
Summer term 2013

Exercises 3, Solutions to tasks 5-8

5. Write Java expressions for the following mathematical expressions:

(a) $\dfrac{a}{b+\dfrac{1}{c}} + 2.5 \cdot 10^6$

```
a / (b + 1/c) + 2.5e6
```

(b) $e^{2k} \cdot \sqrt{x^2 - 2xy + 1}$

```
Math.exp(2*k) * Math.sqrt(x*x - 2*x*y + 1)
```

(c) $z = \begin{cases} 1 & \text{if } n \text{ is even} \\ 0 & \text{otherwise} \end{cases}$

(Remark: $\sqrt{x}$ is `Math.sqrt(x)`, e^x is `Math.exp(x)`,
`a % b` gives the rest when dividing `a` by `b`.)

```
z = (n % 2 == 0 ? 1 : 0);
```

or:

```
if (n % 2 == 0)
   z = 1;
else
   z = 0;
```

or:

```
z = 1 - (n % 2);
```

6.(a) Which errors can possibly occur during runtime of the following Java program fragment?

```
int i;
float list[300];
float x, y;
...
/*  i, x and y are somehow calculated */
...
list[i] = 1.5 / (x + y);
...
```

- `i` can exceed the upper or the lower bound of the indices of the array
- division by 0

(b) Which conditions (to be specified in Java syntax) should be checked to capture these errors before they can cause trouble?

```
(i < 0 || i >= 300) || (x+y == 0)
```

7. The following Java method **f** gets an integer array **x** and the length **n** of the array as arguments:

```
public int f(int x[], int n)
   {
   int i, k = 0;
   if (n <= 0) return -1;
   i = 1;
   while (i < n)
      {
      if (x[k] > x[i])
          k = i;
      i = i+1;
      }
   return k;
   }
```

(a) What does the method **f** calculate?

The index where the minimal entry of array **x** can be found.

(b) What does it give as result if all fields of the array **x** contain the same number, namely, 1 ?

0

8. Write an XL (or Java) program which prints all prime numbers between 1 and 1000 on the screen (and no other numbers).

Remark 1: An integer is a prime number if it is larger than 1 and if it is not divisible without rest by any other positive integer except 1 and itself.
Remark 2: **a % b** = rest of the division of integer **a** by integer **b** ($0 \leq$ **a % b** $<$ **b**).

```
protected void init()
   {
   int UPPER_LIMIT = 1000;
   int n;
   int d;
   boolean p;
   for (n = 2; n <= UPPER_LIMIT; n++)
      {
      p = true;
      d = 2;
      while (p && (double) d <= Math.sqrt((double) n) )
         {
         if (n % d == 0)
            p = false;
         d++;
         }
      if (p) println(n);
      }
   println("finished.");
   }
```

(The operator "(double)" transforms the subsequent integer number into a floating-point number with double precision, to make it accessible to the square-root function, **Math.sqrt**.)