# X1: Scheme basics

Please submit your answers (as a *.scm*, *.ss* or *.txt* file) to marcin.kruczyk@icm.uu.se within a week (September 9).

**Task 1: Introduction**

What value is returned by evaluating each of the following expressions (in order)?

```
1.        2
2.        12.1234
3.        (+ 2 1)
4.        (+ 2 1.0)
5.        (+ (* 2 3) 4)
6.        (define age-of-adam 23)
7.        (define age-of-eva 24)
8.        (> age-of-adam age-of-eva)
9.        (define x 1.4142)
10.       (define y (* x x))
11.       y
12.       (+ x y)
13.       +
14.       (+)
15.       (lambda (a b) (+ (* a a) (* b b)))
16.       ((lambda (a) (+ a a)) 5)
17.       ((lambda (x) (* 2 x)) x)
18.       (define (foo arg)
               (+ arg 3))
19.       (foo)
20.       (define (fee)
               (+ x 5)
               x)
21.       (fee)
22.       (define (fi arg)
               (* 2 arg)
               (+ 3 arg))
23.       (fi)
```

**Task 2: Lambda expressions**

Test the following lambda expressions by applying them to different arguments. What are the results? Explain in words what they do.

```
1.        (lambda (x) x)
2.        (lambda () 2)
3.        (lambda (a b) (+ a b))
4.        (lambda (a b) ((lambda (a) (+ b a)) (+ a 1)))
```

Translate the following mathematical formulae to lambda expressions. Apply the lambda expressions on different values.

1. $\sqrt{x^2}$
2. $\frac{b*h}{2}$
3. $\sqrt{a^2+b^2}$
4. $celsius*1.8+32$

**Task 3: Fahrenheit**
In the previous exercise, you defined a lambda expression that converted degrees in Celsius to Fahrenheit. Create a procedure that does the opposite, i.e. converts from Fahrenheit to Celsius.

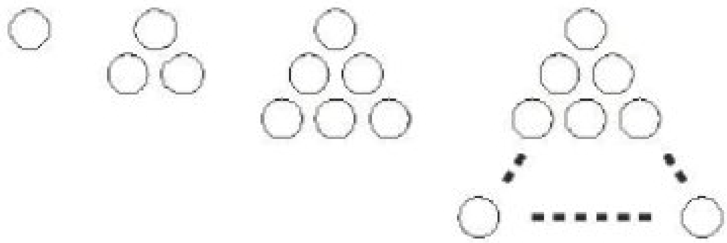
**Procedure:**

```
fahrenheit->celsius: number -> number
```

**Example:**

```
(fahrenheit->celsius 200)
;Value: 93.33333333333333
(fahrenheit->celsius 100)
;Value: 37.77777777777778
(fahrenheit->celsius 70)
;Value: 21.11111111111111
```

**Task 4: How many pins?**
In bowling one often uses ten pins positioned on four rows. How many pins are needed for five rows, six rows, or n rows (where n is a positive integer)?



1. Write a procedure that calculates the number of pins needed for n rows. The procedure should generate a recursive process. Name it `number-of-pins-rec`.
2. Same task as above, but the procedure should generate an iterative process. Name it `number-of-pins-it`.
3. Use the substitution model to show the evaluation of the following two expressions:
   - `(number-of-pins-rec 4)`
   - `(number-of-pins-it 4)`

**Procedure:**

```
number-of-pins-x: number -> number
```

**Example:**

```
(number-of-pins-x 1000)
;Value: 500500
```

**Task 5: Exponentiation**
Write a recursive procedure

```
my-expt: number x number -> number
```

that takes two arguments; a base and a number, and returns the base b raised to the power of the number n, i.e. b^n.

Using the following hint; $b^{2n} = (b^2)^n$, write a new recursive procedure called `fast-expt`.

**Task 6: Testing for primality**
There exist many different procedures for testing primality of numbers. One way to test if a number is a prime is to find the number's divisors. If a number n is prime then n equals the smallest integral divisor of n.

**Implement a procedure that tests if a number is a prime based on the above method.**

Another method is related to Fermat's little theorem:
*If n is a prime number and a is any positive integer less than n, then a raised to the nth power is congruent to a modulo n.*
Two numbers are said to be congruent modulo n if they both have the same remainder when divided by n. Trying a random number a < n, one can be sure that n is not prime if the remainder of a^n modulo n is not equal to a. However, the opposite does not always hold, i.e. a number n is not always prime if the remainder of a^n modulo n is equal to a. By trying more and more random a < n, one can get more confident that n is prime. This algorithm is known as the Fermat test.

**Implement a Fermat test procedure.**

Hints:
Use primitive procedures such as *random*, *modulo* and *remainder*.
Try to abstract the different parts of the methods into primitive procedures.