Functional and Logical Programming

Exercise 1 – Basic Scheme

General Guidelines:

Submission deadline is **Wednesday, November 05, 23:55**
Submit your answers as a single RKT file named ex1-YourID.rkt,
for example: ex1-012345678.rkt
Post the plain-text RKT file in the submission page in course website.
Do NOT pack the file as an archive (no ZIP/RAR/anything).
Do not submit any additional file, or use any other file format.

- No late submission will be accepted! (Submission page will automatically close)
- You should work on your exercise by yourself. Misconducts will be punished harshly.
- Place a comment with your ID at the top your code file.
- Unless specifically noted, you may assume that the input is always correct
  (Your functions should not check for input parameters validity).
- You must **never** change the interface of the functions!

Part 1 – Prefix and Infix (24 points)

A. Translate the following prefix notation expressions to infix notation:
   Example: Given (+ 7 (* 3 4)), the answer is (7 + (3 * 4))
   I. (+ (+ 300 11) (* 8 9))
   II. (+ (* (+ 1 2) 8) (/ 33 11))
   III. (* (* (* (* (* (* (* 7 6) 8) 5) 4) 3) 2) 1)

B. Translate the following infix notation expressions to prefix notation:
   I. (5 + ((8 / 2) - (8 * 9)))
   II. ((8 + 4 + 5) * ((9 + 7) / (8 * 7 * 6)))
   III. (2 – (4 + 8 * 9) / 4)

(Do not compute the values of these expressions. Just translate their notation)
All expressions must be binary (i.e. each operator must have exactly two operands).
Write the answers to this question as comments in your code file, so it will be clear for the grader to see and grade them.

Part 2 – Simple Scheme (26 points)

A. (13 points)
Write the function **`(positiveOdd x)`**
The function receives a number `x`, and returns the symbol '`yes` if `x` is positive and odd, or the symbol '`no` otherwise.

Examples:

`(positiveOdd 4)`	returns '`no`
`(positiveOdd -5)`	returns '`no`
`(positiveOdd 5)`	returns '`yes`

Hint: use `if` or `cond`
Hint: use the function `modulo`

B. (13 points)
Write the function **`(circle-area r)`**
The function receives a radius `r` and returns the circumference of a circle with such radius.
(If you do not remember, the area of a circle with radius `r` is πr^2)

Use the built-in predefined `pi` value
Not a hint, you must use it.

<u>Part 3 – Simple Recursion</u> (25 points)

Write the function `(someSequence n)` that computes the nth value of the following function:

$$f(n) = \begin{cases} f(n-1) + (n * 2)^n & n > 1 \\ 1 & n = 1 \end{cases}$$

(You may assume that `n` is a positive integer)

For example, (someSequence 4) will return 4329
The sequence starts at $n = 1$: $\{1, 17, 233, 4329, 104329 \dots\}$

<u>Hint</u>: use the `expt` function

<u>Part 4 – Less Simple Recursion</u> (25 points)

Fibonacci series is a series of numbers, starting with $1, 1$ in which each element is defined as:

$$a_n = a_{n-2} + a_{n-1}$$

For example, the Fibonacci series for $n = 5$ is $1, 1, 2, 3, 5$.
Write the function `(fibo n)` that prints to the screen the elements of the Fibonacci series for the given parameter `n`.

Example:
```
> (fibo 5)
1
1
2
3
5
```

<u>Hints:</u>
- Use `(display ...)` to print values/strings to screen
- Use `(newline)` to print a line break
- You may use `(begin ...)` or `(let ...)` to execute multiple function calls one after each other
- You should use a helper internal function
- Remember that when writing multiple values one after each other, only the last value is the return value of the expression