Data-types in Erlang

First of all, we have to learn what basic data-types exist in Erlang. Find a description of each of those: Number, Atom, Variable, Tuples, Lists.

Question 1

a) Can an atom contain one or more spaces ?
b) Can an atom start with a capital letter ?
c) Can a list contain elements of different types ?
d) How are strings represented ?

Erlang, a functional language

Unlike C, Erlang is a fully functional language. This means that the all the logic is done through pattern matching and recursion.

Matching can be done on structure (e.g. empty lists), numeric values, atoms and even variables. Matching can also be used to affect values to unused variable. Note that as variable can be modified (they are in fact constant), matching on a variable has very different behaviour depending on whether or not a this variable is used or not.

Matching on structure:

\[
\begin{align*}
\text{length}([\ ]) & \rightarrow 0; \\
\text{length}([\_|T]) & \rightarrow \text{length}(T) + 1.
\end{align*}
\]

Matching on numeric value:

\[
\begin{align*}
isnull(0) & \rightarrow \text{true}; \\
isnull(0.0) & \rightarrow \text{true}; \\
isnull(\_ ) & \rightarrow \text{false}.
\end{align*}
\]

Matching on atoms:

\[
\begin{align*}
\text{receive} \\
\{\text{PID}, \text{ping}\} & \rightarrow \text{PID}!\text{pong}; \\
\{\text{PID}, \text{pong}\} & \rightarrow \\
& \quad \text{io:fwrite("pong from -p!-n", [\text{PID}])}
\end{align*}
\]

End.

Figure 1: Example of matching in erlang

Question 2

a) Write a function \texttt{max(L)} which returns the maximum element of the list \texttt{L}.

b) Write a function \texttt{perimeter(Form)} which computes the perimeter of the form. A form can be any of the following:

- \{\text{square}, Side\}
- \{\text{circle}, Radius\}
- \{\text{triangle}, A, B, C\}

c) Write a function \texttt{sort(L)} which sort \texttt{L}, a list of integer.
Functions are objects!

**Question 3**

a) Write a function \( \text{map}(F, L) \) which returns \([F(L_1), F(L_2), \ldots, F(L_n)]\) with \(L=[L_1, L_2, \ldots, L_n] \).

b) Write a function \( \text{partial}(F, P_1) \) where \( F \) is a two-parameter function. \( \text{partial}(F, P_1) \) must return a one parameter function which, if given \( P_2 \), would return \( F(P_1, P_2) \).

c) Use \( \text{partial} \) and \( \text{map} \) to write \( \text{multimap}(F, L) \) which does the same as \( \text{map} \) but takes and returns a list of lists.

**Spawning and communication for distributed systems**

Erlang provides three main primitives required for concurrency: spawning of new processes, sending of messages and message receipt.

**Question 4**

a) Write a function \( \text{multispawn}(L) \) where \( L \) is a list of tuples of the form \( \{M, F, L\} \) where:
   - \( M \) is a module name (an atom)
   - \( F \) is a function name (an atom)
   - \( L \) is a parameter list for \( F \)

\( \text{multispawn} \) must, for each tuple, spawn one process which computes \( M:F(L) \). Multispawn must return the list of PIDs for those spawn processes.

b) Write a function \( \text{dreturn}(\text{Pid}, T) \) that takes a \( \{M,F,L\} \) tuple \( T \) as above, computes \( M:F(L) \) and then sends back \( \{\text{Result}, \text{self()}\} \) to \( \text{Pid} \).

c) Write a \( \text{delegate}(L) \) which is the same as \( \text{multispawn} \) but returns a list of results (rather then a list of spawned PIDs).

**Distributed algorithms**

**Question 5**

a) Write a function \( \text{dmax}(L) \) which computes a distributed maximum
   - \( L \) is a list of integers
   - if the list is bigger then a certain length then
     a) split the list into smaller sub-lists
     b) spawn one process for each sub-lists and make this process compute the maximum element of the sublist and send it back
     c) gather the results in a list
     d) compute the maximum of that list

b) (Bonus) Using the same approach, write an efficient distributed sort \( \text{dsort} \). Benchmark the results.