Linguagens de Programação (CK0115)
Lista de Exercícios III (Capítulo 3)

Fernanda Costa de Sousa - 485404

1. Absolute value of real numbers.

Não funciona pois ao invés de 0, o correto seria 0.0.

2.Raiz cúbica

```
declare
fun {Cubert X}
  fun {CubertIter Guess}
     if {GoodEnough Guess} then Guess
     else {CubertIter {Improve Guess}} end
  end
  fun {GoodEnough Guess}
     {Abs X-Guess*Guess*Guess}/X < 0.000001
  end
  fun {Improve Guess}
     (X/(Guess*Guess) + 2.0*Guess)/3.0
  end
  Guess = 1.0
in
  {CubertIter Guess}
end
{Browse {Cubert 3.0}}
```

3. The half-interval method

```
declare
fun {BscMethod F A B}
  X = (A+B)/2.0
  V = {F X}
in
  if {Abs V} < 0.0001 then X
  else
     if V > 0.0 then {BscMethod F A X}
     else {BscMethod F X B} end
  end
end
```

```
declare
fun {BscMethod F A B}
  fun {BscMethodIter A B}
     X = (A+B)/2.0
     V = {F X}
  in
     if {GoodEnough V} then X
     else A1 B1 in
    A1#B1={Improve A B V X}
    {BscMethodIter A1 B1}
     end
  end
  fun {GoodEnough V}
     {Abs V} < 0.00001
  end
  fun {Improve A B V X}
     if V > 0.0 then A#X
     else X#B end
  end
in
  {BscMethodIter A B}
end

{Browse {BscMethod fun{$ X} X*X - 2.0 end 0.0 2.0}}
```

4. Iterative factorial

```
declare
fun {Fact N}
  fun {FactIterative N A}
     if N==1 then A
     else {FactIterative N-1 N*A} end
  end
in
  {FactIterative N 1}
end

{Browse {Fact 10}}
```

5. An iterative SumList

```
declare
fun {SumList Xs}
  fun {SumListIterative Ys A}
     case Ys
     of nil then A
     [] Y|Yr then {SumListIterative Yr A+Y} end
  end
in
  {SumListIterative Xs 0}
end

{Browse {SumList [1 2 3 4 5]}}
```

7. Another append function

```
fun {CorrectAppend Ls Ms}
   case Ls
   of nil then Ms
   [] L|Lr then L|{Append Lr Ms}
   end
end
fun {Append Ls Ms}
   case Ms
   of nil then Ls
   [] X|Mr then {Append {Append Ls [X]} Mr}
   end
end
```

8. An iterative append

```
fun {Append Xs Ys}
  fun {ReverseAppendIter Xs Ys}
     case Xs
     of nil then Ys
     [] X|Xr then {ReverseAppendIter Xr X|Ys} end
  end
  fun {Reverse Xs}
     fun {ReverseIter Xs A}
   case Xs
   of nil then A
```

```
   [] X|Xr then {ReverseIter Xr X|A} end
     end
  in
     {ReverseIter Xs nil}
  end
in
  {ReverseAppendIter {Reverse Xs} Ys}
end
```

10. Checking if something is a list.

```
fun {Leaf X} X\=(_|_) end
```

essa parte poderia ser reescrita da seguinte maneira:

```
fun {Leaf X} if X==(_|_) then false else true end end
```

Se usarmos essa versão da lista seria ruim, pois ocorreria um bloqueio. A instrução *case* pode ter a mesma estrutura mas ao verificar na inclusão, a comparação com o valor indefinido é um fator que bloqueia.

11. Limitations of difference lists

Fazer um Append em uma lista diferença, seria o mesmo que colocar o final dessa lista em outra. Não é possível fazer isso mais de uma vez. Assim como é descrito no livro texto: *Essa propriedade significa que as listas diferença só podem ser usadas em circunstâncias especiais (pág. 142).*