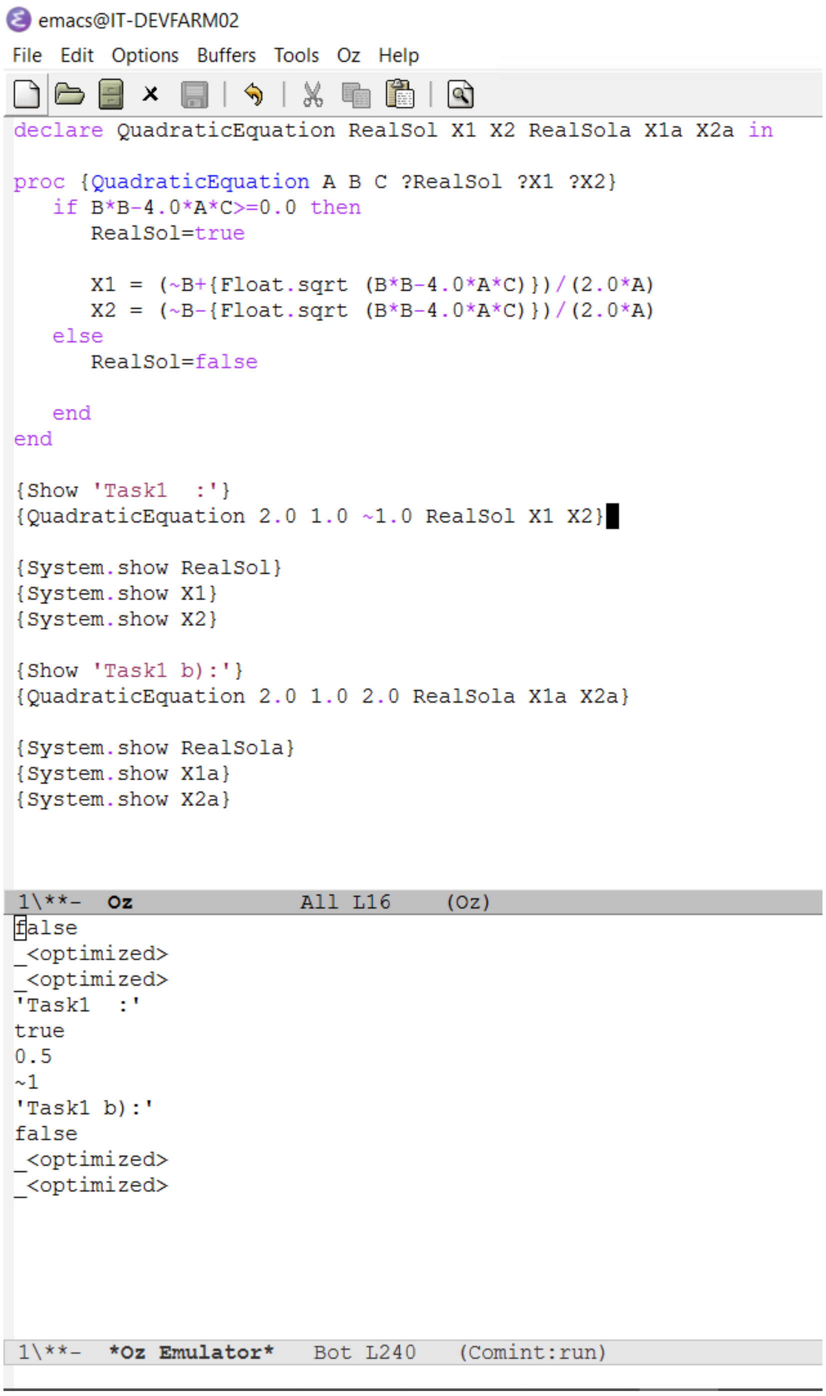
Javier Fernández Rozas – Oz

Task1



```
declare QuadraticEquation RealSol X1 X2 RealSola X1a X2a in

proc {QuadraticEquation A B C ?RealSol ?X1 ?X2}
   if B*B-4.0*A*C>=0.0 then
      RealSol=true

      X1 = (~B+{Float.sqrt (B*B-4.0*A*C)})/(2.0*A)
      X2 = (~B-{Float.sqrt (B*B-4.0*A*C)})/(2.0*A)
   else
      RealSol=false

   end
end

{Show 'Task1  :'}
{QuadraticEquation 2.0 1.0 ~1.0 RealSol X1 X2}

{System.show RealSol}
{System.show X1}
{System.show X2}

{Show 'Task1 b):'}
{QuadraticEquation 2.0 1.0 2.0 RealSola X1a X2a}

{System.show RealSola}
{System.show X1a}
{System.show X2a}
```

```
1\**-  Oz                 All L16     (Oz)
```

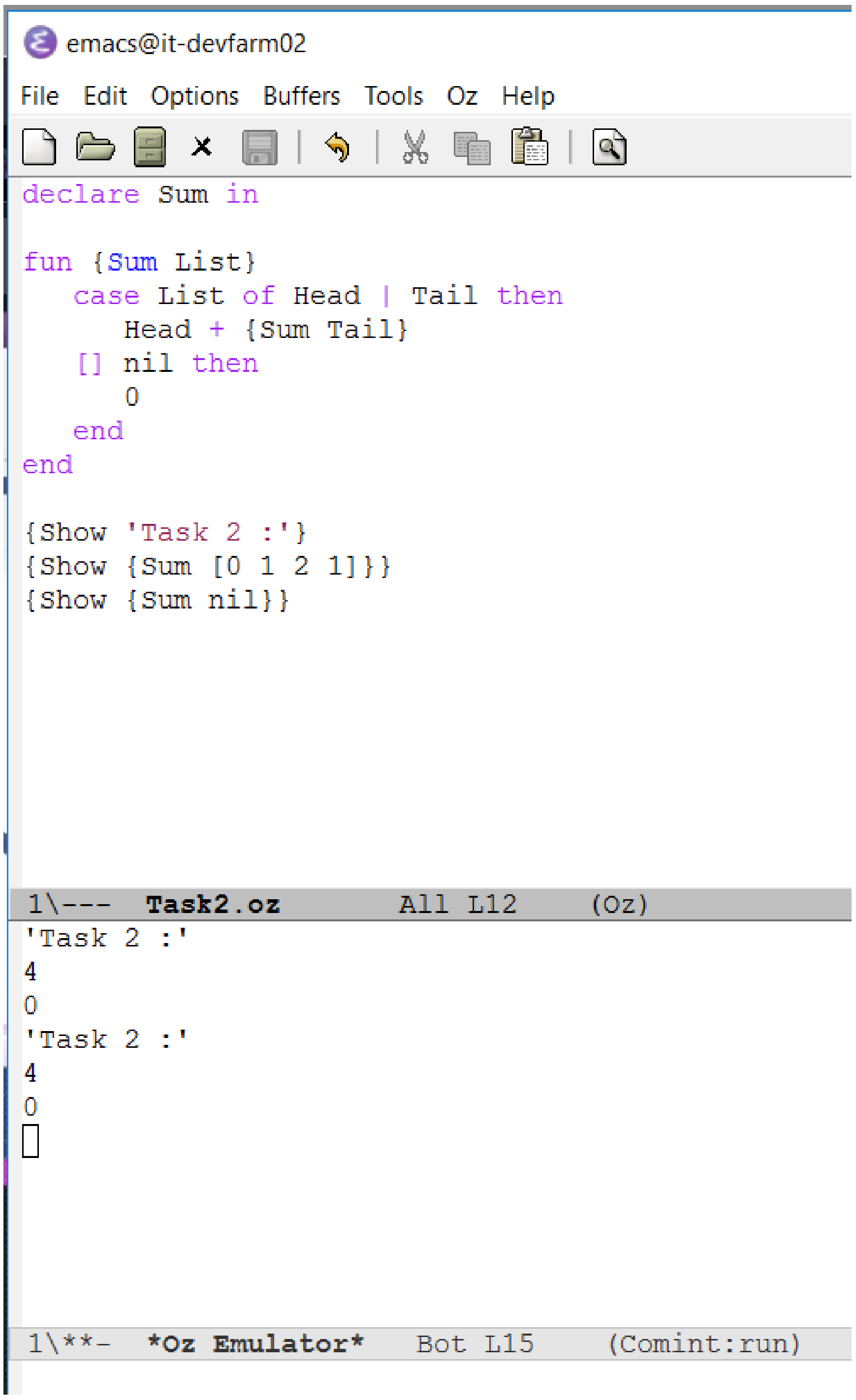
```
false
_<optimized>
_<optimized>
'Task1  :'
true
0.5
~1
'Task1 b):'
false
_<optimized>
_<optimized>
```

```
1\**-  *Oz Emulator*    Bot L240    (Comint:run)
```

C) They let us focus on higher level design (hiding non-important details), making code easier to read/understand

D)Procedures don't have to return an explicit value. In Oz, functions are an specific case of procedures

Task 2



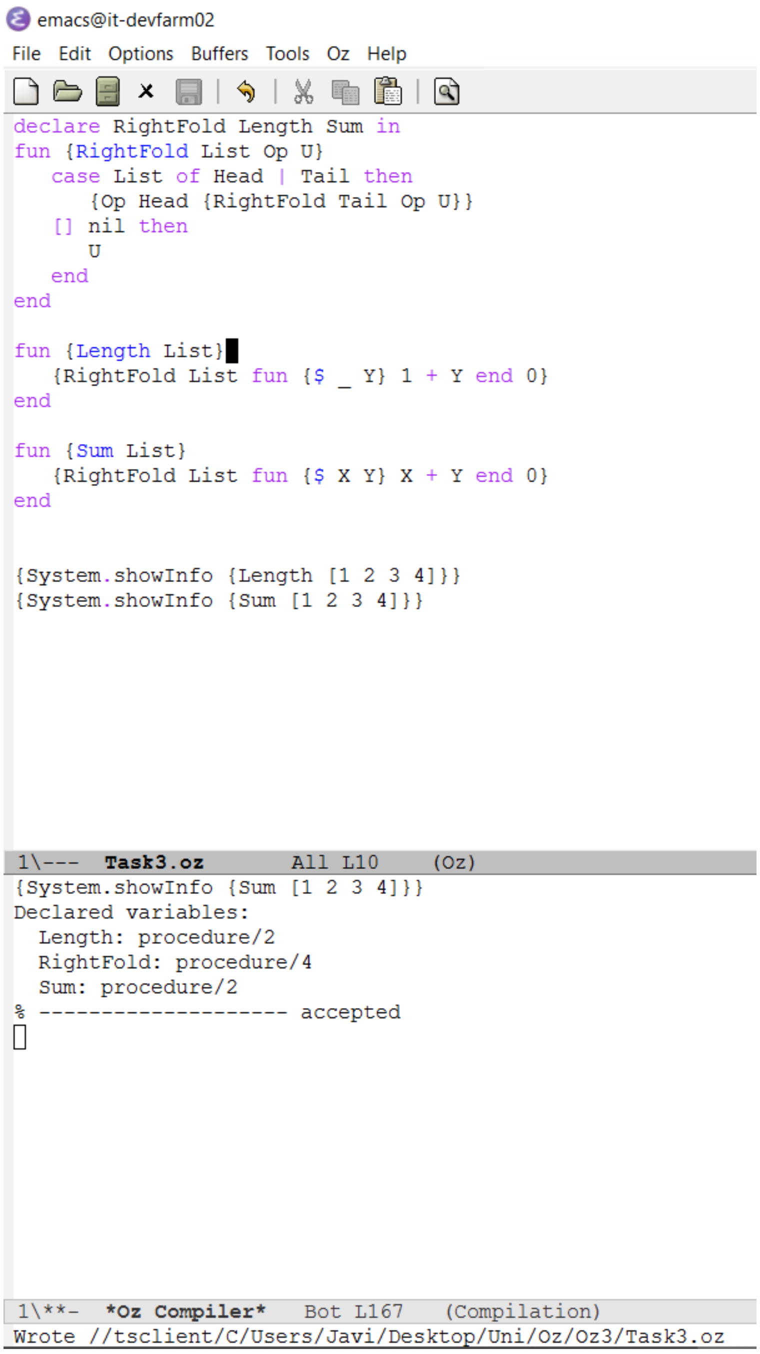
```
declare Sum in

fun {Sum List}
   case List of Head | Tail then
      Head + {Sum Tail}
   [] nil then
      0
   end
end

{Show 'Task 2 :'}
{Show {Sum [0 1 2 1]}}
{Show {Sum nil}}
```

```
 1\---  Task2.oz        All L12      (Oz)
```

```
'Task 2 :'
4
0
'Task 2 :'
4
0
```

```
 1\**-  *Oz Emulator*    Bot L15      (Comint:run)
```

Task 3



```
declare RightFold Length Sum in
fun {RightFold List Op U}
   case List of Head | Tail then
      {Op Head {RightFold Tail Op U}}
   [] nil then
      U
   end
end

fun {Length List}
   {RightFold List fun {$ _ Y} 1 + Y end 0}
end

fun {Sum List}
   {RightFold List fun {$ X Y} X + Y end 0}
end


{System.showInfo {Length [1 2 3 4]}}
{System.showInfo {Sum [1 2 3 4]}}
```

```
1\---  Task3.oz        All L10     (Oz)
```

```
{System.showInfo {Sum [1 2 3 4]}}
Declared variables:
  Length: procedure/2
  RightFold: procedure/4
  Sum: procedure/2
% -------------------- accepted
```

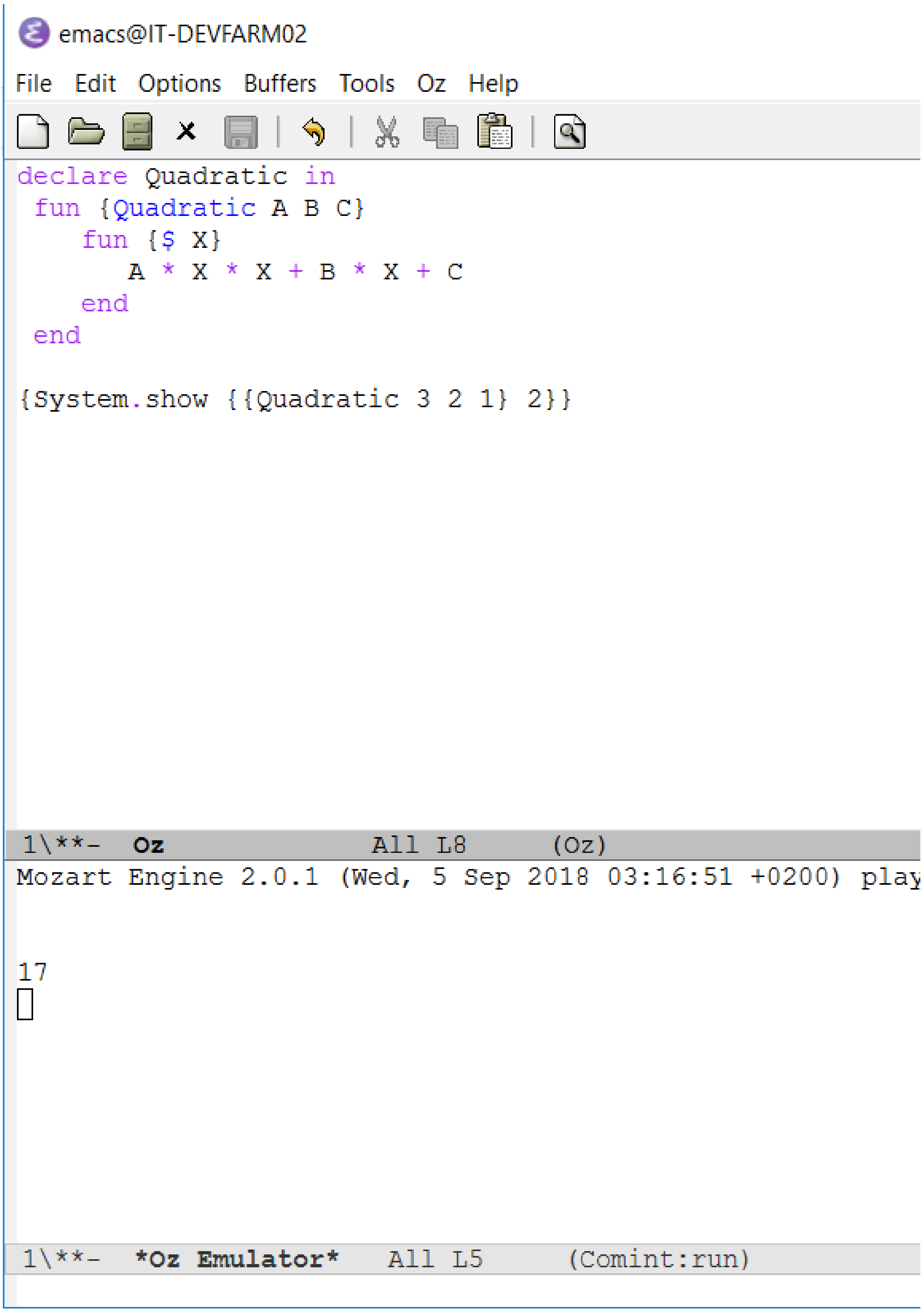
```
1\**-  *Oz Compiler*   Bot L167    (Compilation)
Wrote //tsclient/C/Users/Javi/Desktop/Uni/Oz/Oz3/Task3.oz
```

b) 1.Declare function, 2.case of H|T 3.Call the Op with Head and the recursive call to RightFold as the second argument (X Op (Y Op Z))… 4.Case Nil 5.Return "U" (Neutral Operator) when list empty 6.Close function

d)For Sum and Length the result would be the same, because they are associative, but for other operations, such as subtractions, divisions, that are not associative, the result would be different

e)The value of 1, to not alter the multiplication (X * 1= X)

Task 4



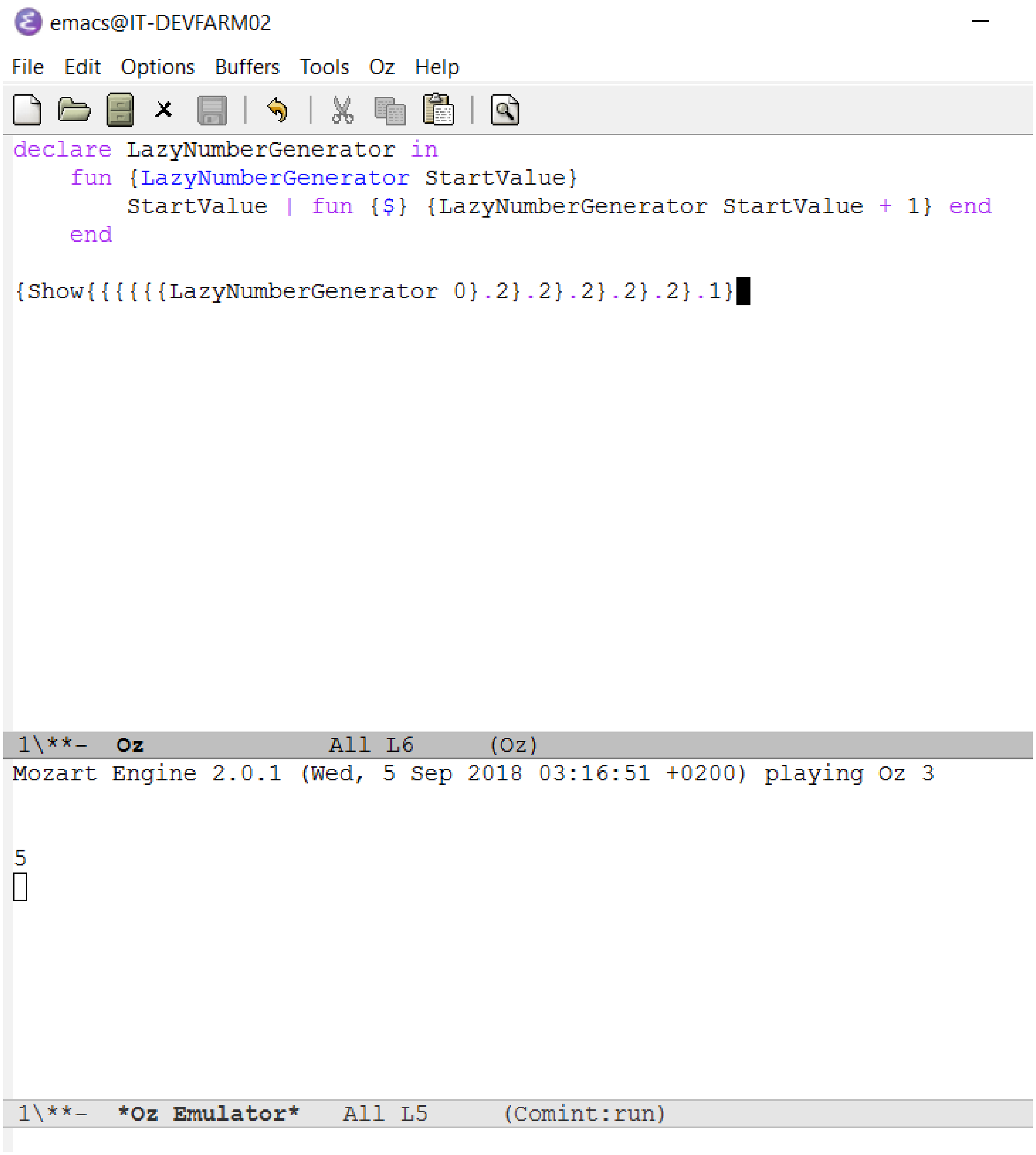
```
declare Quadratic in
 fun {Quadratic A B C}
    fun {$ X}
       A * X * X + B * X + C
    end
 end

{System.show {{Quadratic 3 2 1} 2}}
```

```
1\**-  Oz              All L8       (Oz)
```

```
Mozart Engine 2.0.1 (Wed, 5 Sep 2018 03:16:51 +0200) play


17
```

```
1\**-  *Oz Emulator*   All L5       (Comint:run)
```

Task5



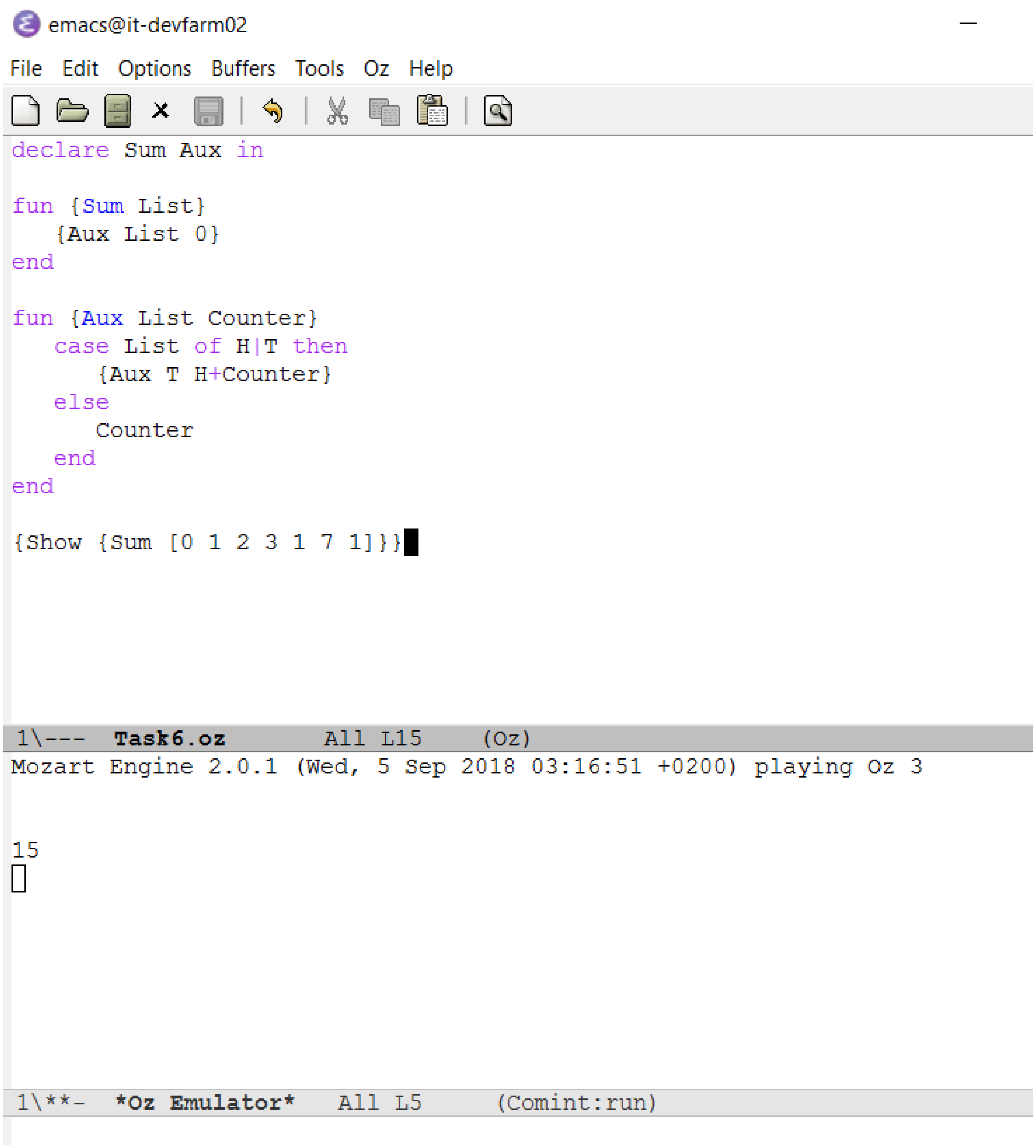
```
declare LazyNumberGenerator in
    fun {LazyNumberGenerator StartValue}
        StartValue | fun {$} {LazyNumberGenerator StartValue + 1} end
    end

{Show{{{{{LazyNumberGenerator 0}.2}.2}.2}.2}.2}.1}
```

```
1\**-  Oz              All L6      (Oz)
```

```
Mozart Engine 2.0.1 (Wed, 5 Sep 2018 03:16:51 +0200) playing Oz 3


5
```

```
1\**-  *Oz Emulator*   All L5      (Comint:run)
```

Task6



```
declare Sum Aux in

fun {Sum List}
   {Aux List 0}
end

fun {Aux List Counter}
   case List of H|T then
      {Aux T H+Counter}
   else
      Counter
   end
end

{Show {Sum [0 1 2 3 1 7 1]}}
```

```
1\---  Task6.oz        All L15     (Oz)
Mozart Engine 2.0.1 (Wed, 5 Sep 2018 03:16:51 +0200) playing Oz 3


15

1\**-  *Oz Emulator*    All L5      (Comint:run)
```

A) No, it was not. This new version implements a counter that is passed each time a recursive call is done, so that value is passed when the recursion end.

B) It puts less resources into the semantic stack, because there is no need to retain all the statements

C)To benefit from Tail recursion, the language/compiler has to support this optimization (replacing stack).If present, the language would benefit from it.