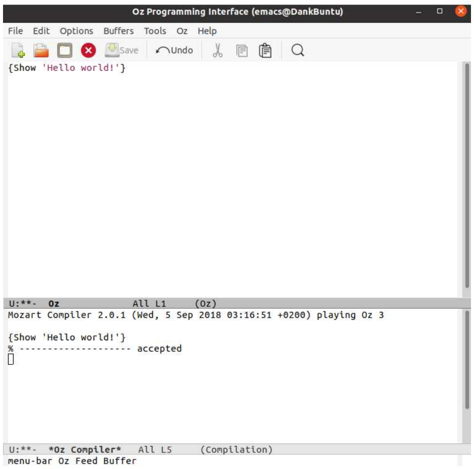
TDT4165 PROGRAMMING LANGUAGES
Assignment 1 Introduction to Oz

Danilas Miscenko

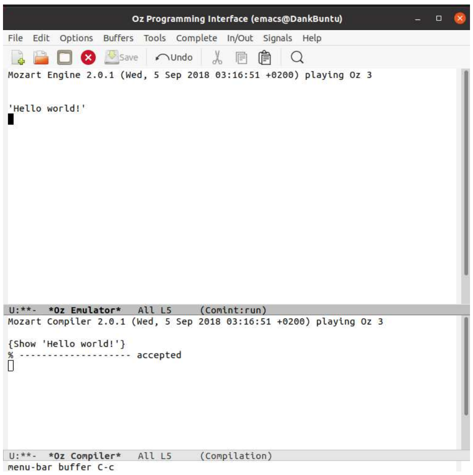
September 14, 2020

Task 1: Hello world!


Oz Programming Interface (emacs@DankBuntu)
File Edit Options Buffers Tools Oz Help
Save
Undo

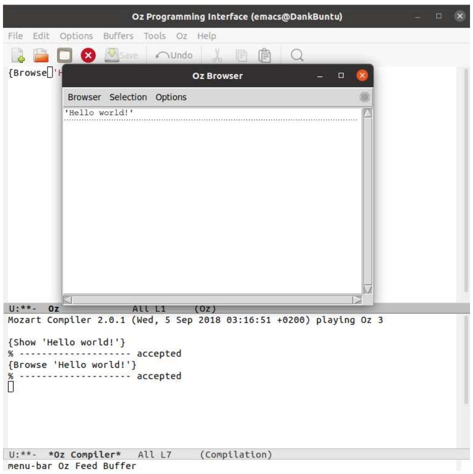
```
{Show 'Hello world!'}

U:**-  Oz              All L1     (Oz)
Mozart Compiler 2.0.1 (Wed, 5 Sep 2018 03:16:51 +0200) playing Oz 3

{Show 'Hello world!'}
% -------------------- accepted

U:**-  *Oz Compiler*   All L5     (Compilation)
menu-bar Oz Feed Buffer
```

Figure 1: The Emacs environment


Oz Programming Interface (emacs@DankBuntu)
File Edit Options Buffers Tools Complete In/Out Signals Help
Save
Undo

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


'Hello world!'

U:**-  *Oz Emulator*   All L5     (Comint:run)
Mozart Compiler 2.0.1 (Wed, 5 Sep 2018 03:16:51 +0200) playing Oz 3

{Show 'Hello world!'}
% -------------------- accepted

U:**-  *Oz Compiler*   All L5     (Compilation)
menu-bar buffer C-c
```

Figure 2: The Show output in Oz Emulator

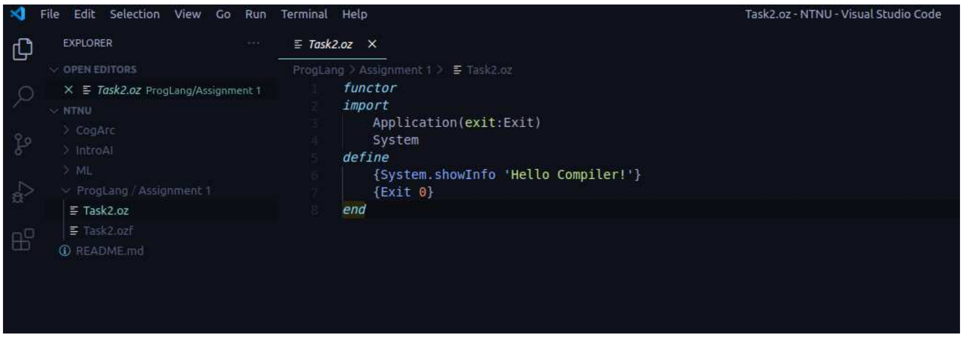

Oz Programming Interface (emacs@DankBuntu)
File Edit Options Buffers Tools Oz Help
Save
Undo
Oz Browser
Browser Selection Options

```
{Browse 'H

'Hello world!'

U:**-  Oz              All L1     (Oz)
Mozart Compiler 2.0.1 (Wed, 5 Sep 2018 03:16:51 +0200) playing Oz 3

{Show 'Hello world!'}
% -------------------- accepted
{Browse 'Hello world!'}
% -------------------- accepted

U:**-  *Oz Compiler*   All L7     (Compilation)
menu-bar Oz Feed Buffer
```

Figure 3: The Oz Browser output

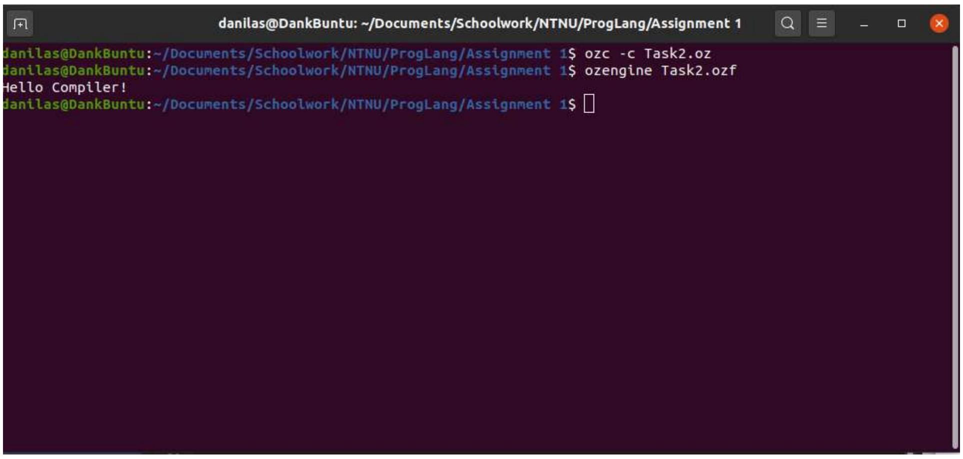
Task 2: Compiling Oz

My favourite editor happens to be Visual Studio Code.


File Edit Selection View Go Run Terminal Help
Task2.oz - NTNU - Visual Studio Code
EXPLORER
OPEN EDITORS
Task2.oz ProgLang/Assignment 1
NTNU
CogArc
IntroAI
ML
ProgLang / Assignment 1
Task2.oz
Task2.ozf
README.md
Task2.oz
ProgLang > Assignment 1 > Task2.oz

```
functor
import
    Application(exit:Exit)
    System
define
    {System.showInfo 'Hello Compiler!'}
    {Exit 0}
end
```

Figure 4: Code in VS Code

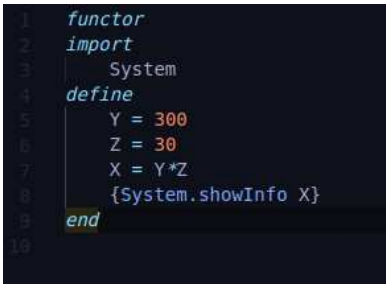

danilas@DankBuntu: ~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1

```
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozc -c Task2.oz
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozengine Task2.ozf
Hello Compiler!
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$
```

Figure 5: Compiling in terminal and running the code

Task 3: Variables

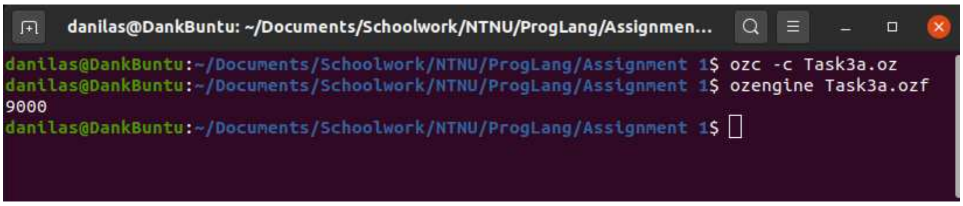
a) As I understood the task, I just declared two extra variables as 300 and 30, and then ”calculated” X out of those.



```
functor
import
    System
define
    Y = 300
    Z = 30
    X = Y*Z
    {System.showInfo X}
end
```

Figure 6: VS code

The output is the variable X, which is still 9000

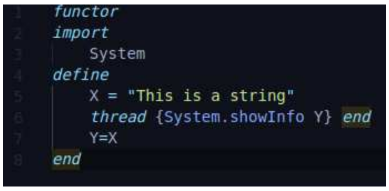


danilas@DankBuntu: ~/Documents/Schoolwork/NTNU/ProgLang/Assignmen...

```
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozc -c Task3a.oz
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozengine Task3a.ozf
9000
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ 
```

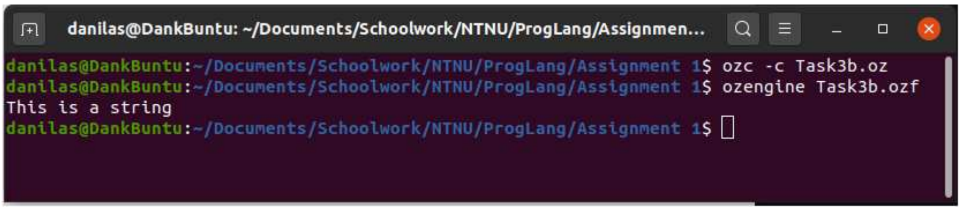
Figure 7: Terminal compiling and output

b) Here's the code:



```
functor
import
    System
define
    X = "This is a string"
    thread {System.showInfo Y} end
    Y=X
end
```

Figure 8: VS code



danilas@DankBuntu: ~/Documents/Schoolwork/NTNU/ProgLang/Assignmen...

```
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozc -c Task3b.oz
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozengine Task3b.ozf
This is a string
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ 
```

Figure 9: Terminal compiling and output

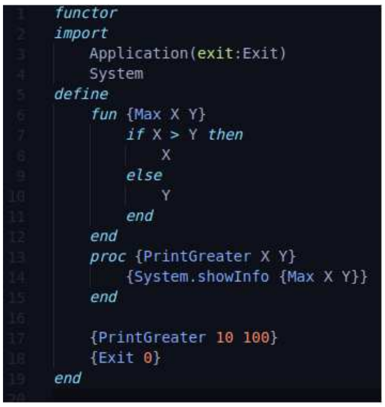
showInfo prints Y before it is assigned because the thread waits until the values used in the thread are assigned. In Oz, threads are designed to be able to communicate with each other through variables.

This behaviour makes multithreading a lot easier and more intuitive to implement.

The statement Y=X assigns Y to be the same value as X, that being the string "This is a string". The thread that uses Y waits until this value is assigned before showing the information of Y.

Task 4: Functions and procedure

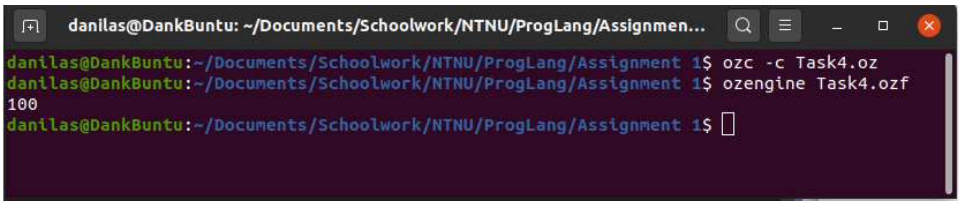
Here’s the code for both the function and the procedure:



```
functor
import
    Application(exit:Exit)
    System
define
    fun {Max X Y}
        if X > Y then
            X
        else
            Y
        end
    end
    proc {PrintGreater X Y}
        {System.showInfo {Max X Y}}
    end

    {PrintGreater 10 100}
    {Exit 0}
end
```

Figure 10: VS Code

The code runs correctly:

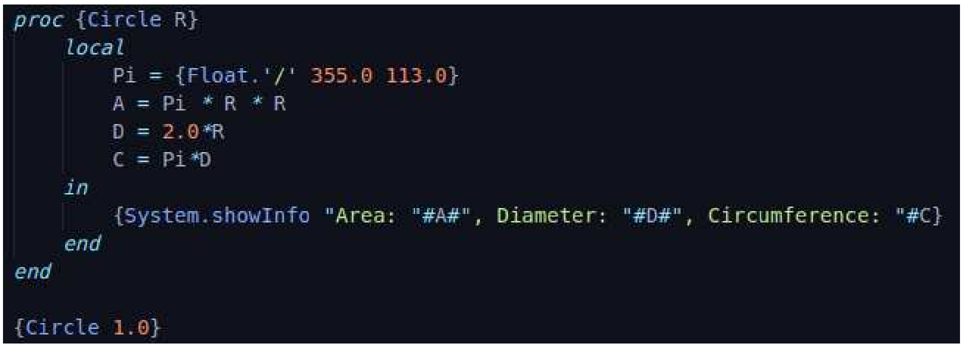


danilas@DankBuntu: ~/Documents/Schoolwork/NTNU/ProgLang/Assignmen...

```
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozc -c Task4.oz
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozengine Task4.ozf
100
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$
```

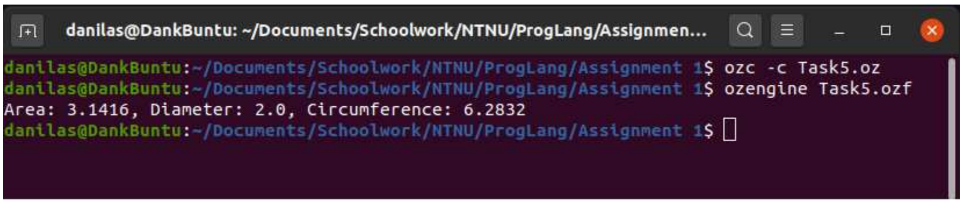
Figure 11: Terminal compiling and output

Task 5: Variables II



```
proc {Circle R}
    local
        Pi = {Float.'/' 355.0 113.0}
        A = Pi * R * R
        D = 2.0*R
        C = Pi*D
    in
        {System.showInfo "Area: "#A#", Diameter: "#D#", Circumference: "#C}
    end
end

{Circle 1.0}
```

Figure 12: VS Code



danilas@DankBuntu: ~/Documents/Schoolwork/NTNU/ProgLang/Assignmen...

```
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozc -c Task5.oz
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozengine Task5.ozf
Area: 3.1416, Diameter: 2.0, Circumference: 6.2832
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ 
```

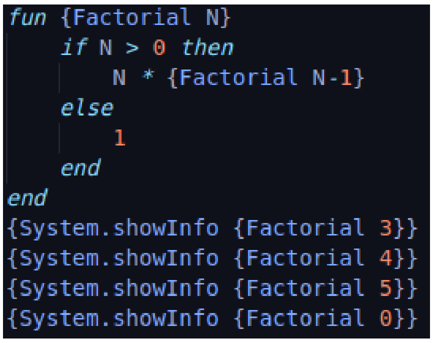
Figure 13: Terminal compiling and output

Task 6: Recursion

I run the Factorial function 4 times. The first three are to check that i get a correct answer, while the fourth is to show that it forks for zero factorial.

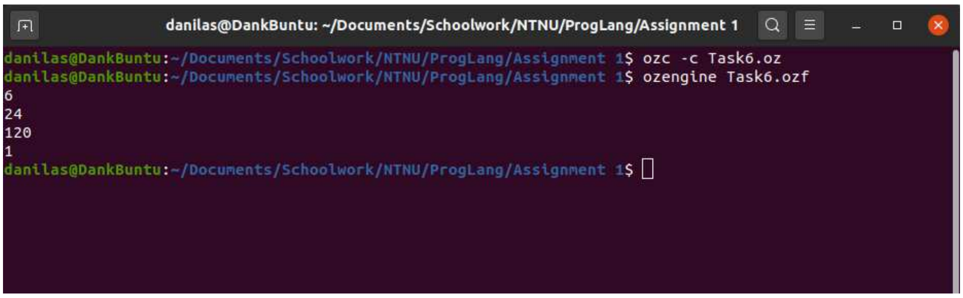
$$3! = 3 * 2 * 1 = 6$$
$$4! = 3! * 4 = 6 * 4 = 24$$
$$5! = 4! * 5 = 24 * 5 = 120$$

Zero factorial is defined to equal to 1.



```
fun {Factorial N}
    if N > 0 then
        N * {Factorial N-1}
    else
        1
    end
end
{System.showInfo {Factorial 3}}
{System.showInfo {Factorial 4}}
{System.showInfo {Factorial 5}}
{System.showInfo {Factorial 0}}
```

Figure 14: VS Code



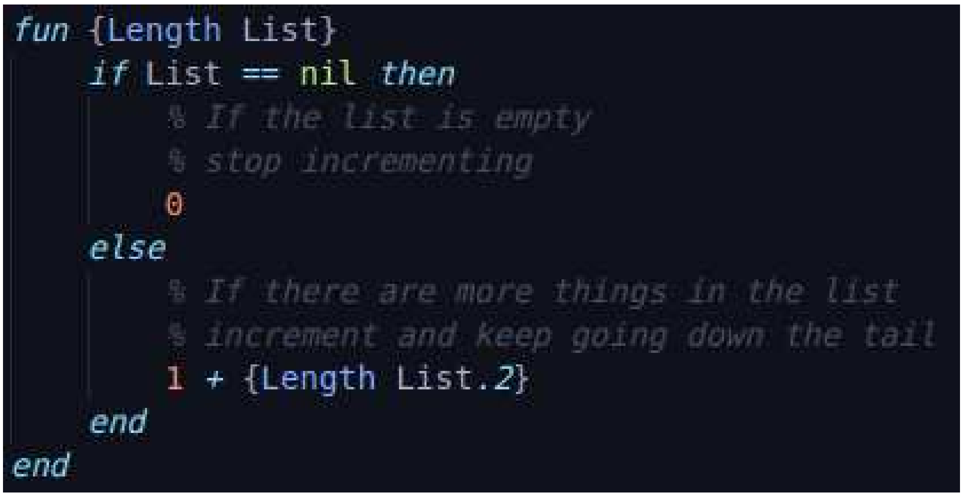
danilas@DankBuntu: ~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1

```
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozc -c Task6.oz
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozengine Task6.ozf
6
24
120
1
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ 
```

Figure 15: Terminal compiling and output

Task 7: Lists

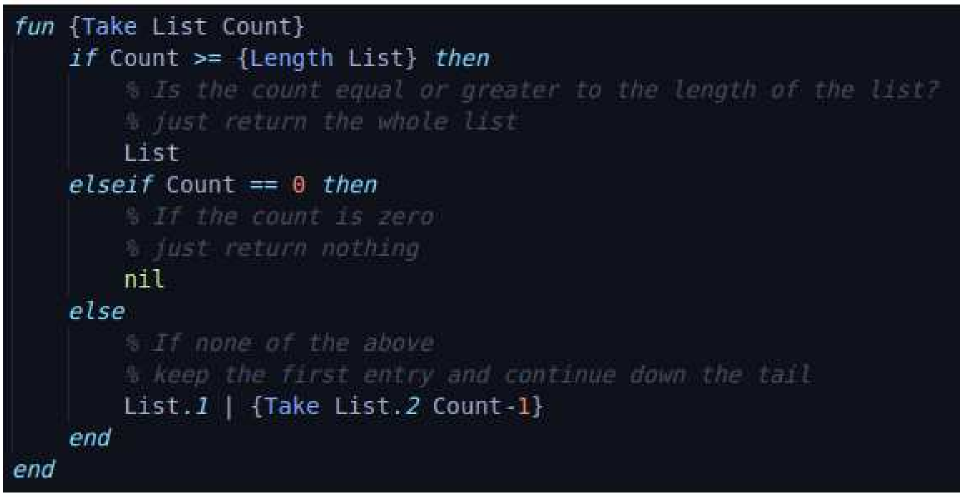
a) The Length function



```
fun {Length List}
    if List == nil then
        % If the list is empty
        % stop incrementing
        0
    else
        % If there are more things in the list
        % increment and keep going down the tail
        1 + {Length List.2}
    end
end
```

Figure 16: Length function

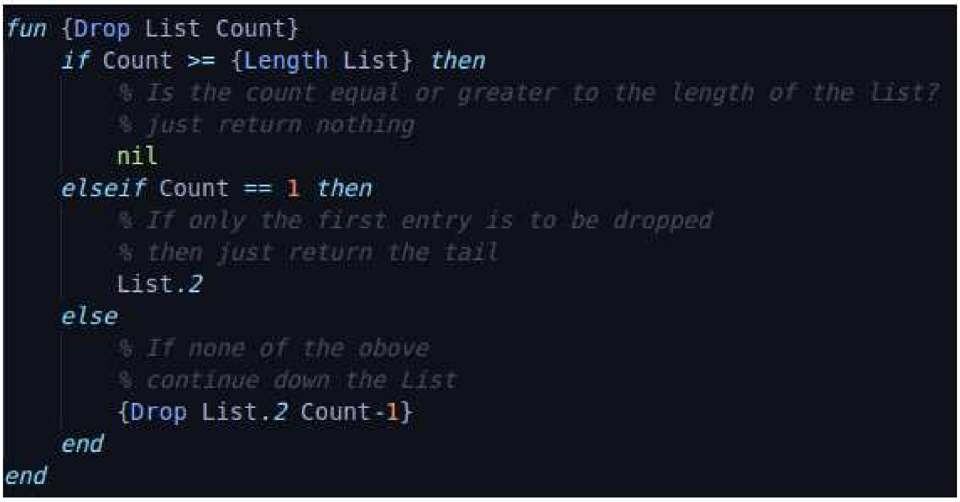
b) The Take function



```
fun {Take List Count}
    if Count >= {Length List} then
        % Is the count equal or greater to the length of the list?
        % just return the whole list
        List
    elseif Count == 0 then
        % If the count is zero
        % just return nothing
        nil
    else
        % If none of the above
        % keep the first entry and continue down the tail
        List.1 | {Take List.2 Count-1}
    end
end
```

Figure 17: Take function

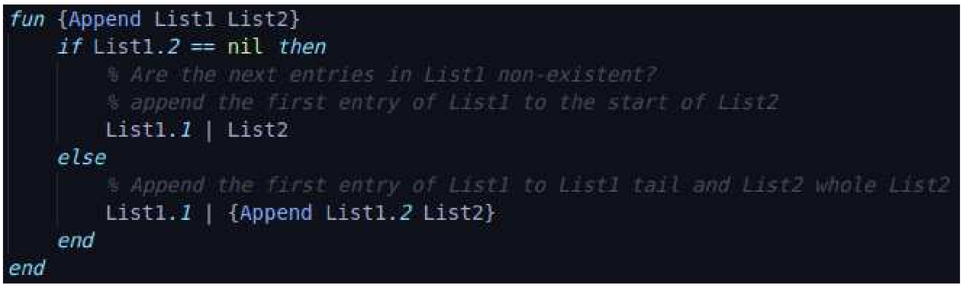
c) The Drop function



```
fun {Drop List Count}
    if Count >= {Length List} then
        % Is the count equal or greater to the length of the list?
        % just return nothing
        nil
    elseif Count == 1 then
        % If only the first entry is to be dropped
        % then just return the tail
        List.2
    else
        % If none of the above
        % continue down the List
        {Drop List.2 Count-1}
    end
end
```

Figure 18: Drop function

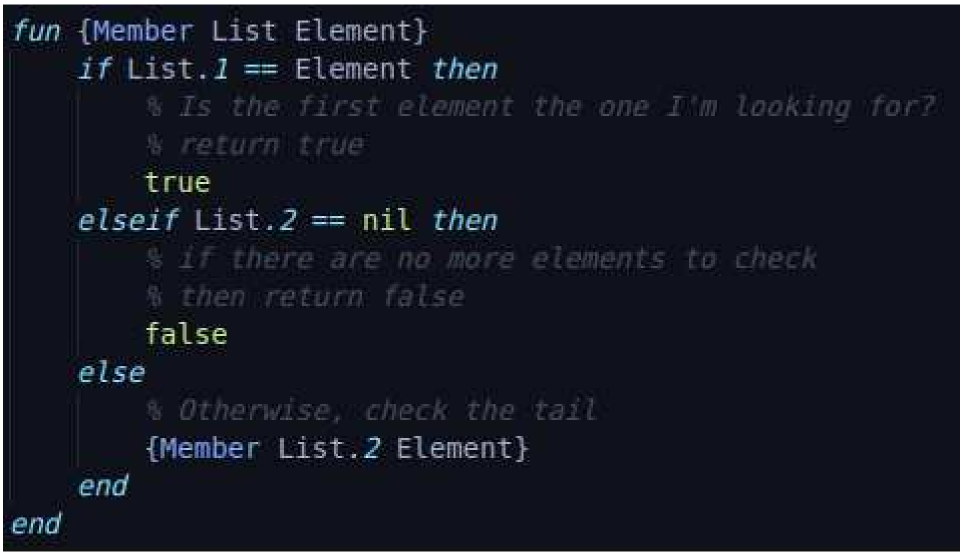
d) The Append function



```
fun {Append List1 List2}
    if List1.2 == nil then
        % Are the next entries in List1 non-existent?
        % append the first entry of List1 to the start of List2
        List1.1 | List2
    else
        % Append the first entry of List1 to List1 tail and List2 whole List2
        List1.1 | {Append List1.2 List2}
    end
end
```

Figure 19: Append function

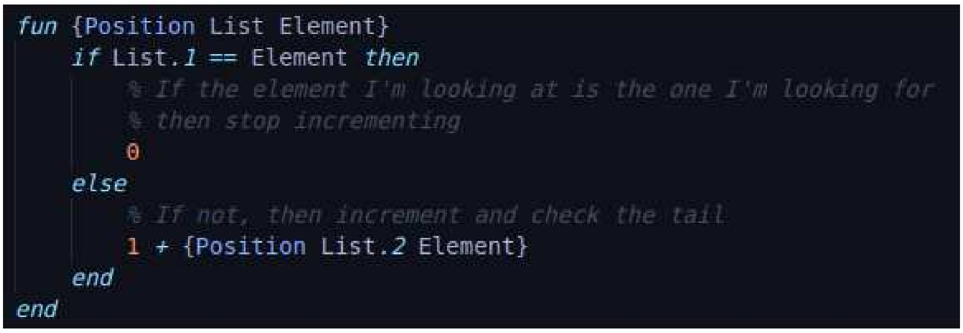
e) The Member function



```
fun {Member List Element}
    if List.1 == Element then
        % Is the first element the one I'm looking for?
        % return true
        true
    elseif List.2 == nil then
        % if there are no more elements to check
        % then return false
        false
    else
        % Otherwise, check the tail
        {Member List.2 Element}
    end
end
```

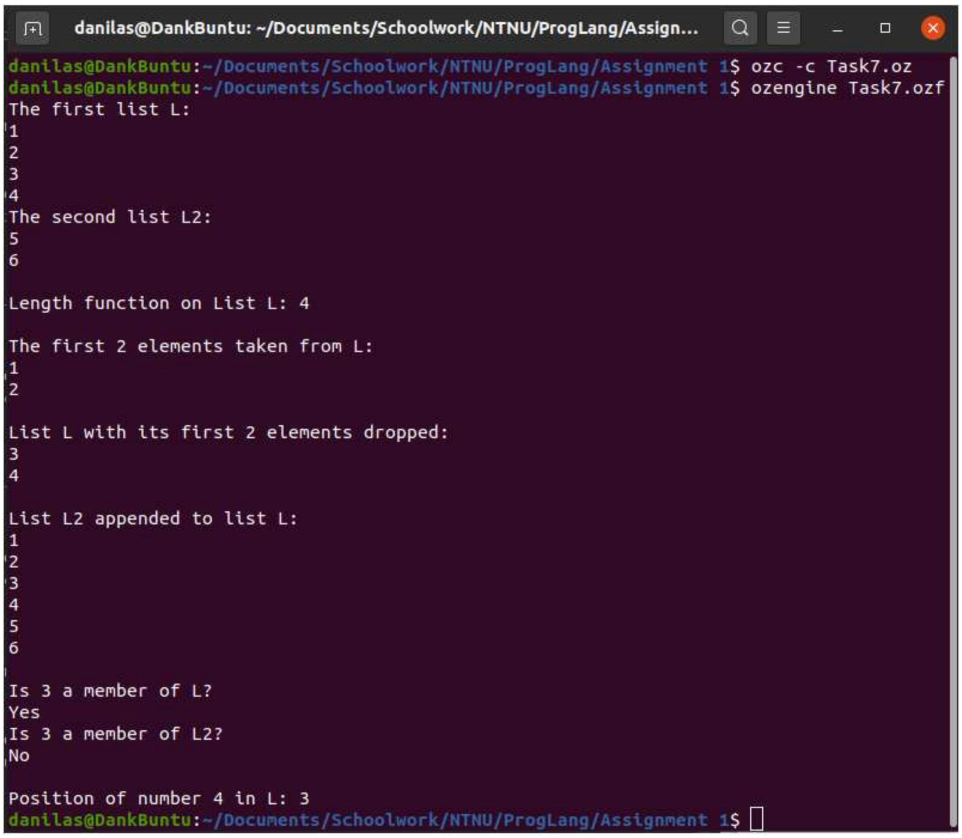
Figure 20: Member function

f) The Position function



```
fun {Position List Element}
    if List.1 == Element then
        % If the element I'm looking at is the one I'm looking for
        % then stop incrementing
        0
    else
        % If not, then increment and check the tail
        1 + {Position List.2 Element}
    end
end
```

Figure 21: Position function



danilas@DankBuntu: ~/Documents/Schoolwork/NTNU/ProgLang/Assign...

```
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozc -c Task7.oz
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ ozengine Task7.ozf
The first list L:
1
2
3
4
The second list L2:
5
6

Length function on List L: 4

The first 2 elements taken from L:
1
2

List L with its first 2 elements dropped:
3
4

List L2 appended to list L:
1
2
3
4
5
6

Is 3 a member of L?
Yes
Is 3 a member of L2?
No

Position of number 4 in L: 3
danilas@DankBuntu:~/Documents/Schoolwork/NTNU/ProgLang/Assignment 1$ 
```

Figure 22: Terminal compiling and output