Programming Paradigms
Seminar 1
Weeks (29 September – 3 October 2025
and
6 October – 10 October 2025)

Seminar Rules

- seminar activity will be done at the group level
groups are fixed by me at the first seminar and they cannot be changed later
– please start the work in class and complete at home until the deadline

DEADLINE:

weeks: 13-17 October 2025 and 20-24 October 2024

Mozart Installation

Please install the Windows binary version mozart2-2.0.1 from https://sourceforge.net/projects/mozart-oz/files/. If you prefer you can also work on Linux.

Mozart programming system is using an Emacs editor. Some of the useful key bindings are enumerated below:

Key Bindings

C-. C-l	Feed current line
C-. C-r	Feed selected region
C-. C-b	Feed whole buffer
C-. C-p	Feed current paragraph
C-. c	Toggle display of *Oz Compiler* buffer
C-. e	Toggle display of *Oz Emulator* buffer
C-x ‘ (i.e. Control-x backquote)	positions the transcript to make the first error message visible and moves the point, in the source buffer, to where the bug is likely to be located.
C-. n	Create a new buffer using the Oz major mode. Note that this buffer has no associated file name, so quitting Emacs will kill it without warning.

M-n
M.p Switch to the previous resp. next buffer in the buffer list that runs in an Oz mode. If no such buffer exists, an error is signalled.

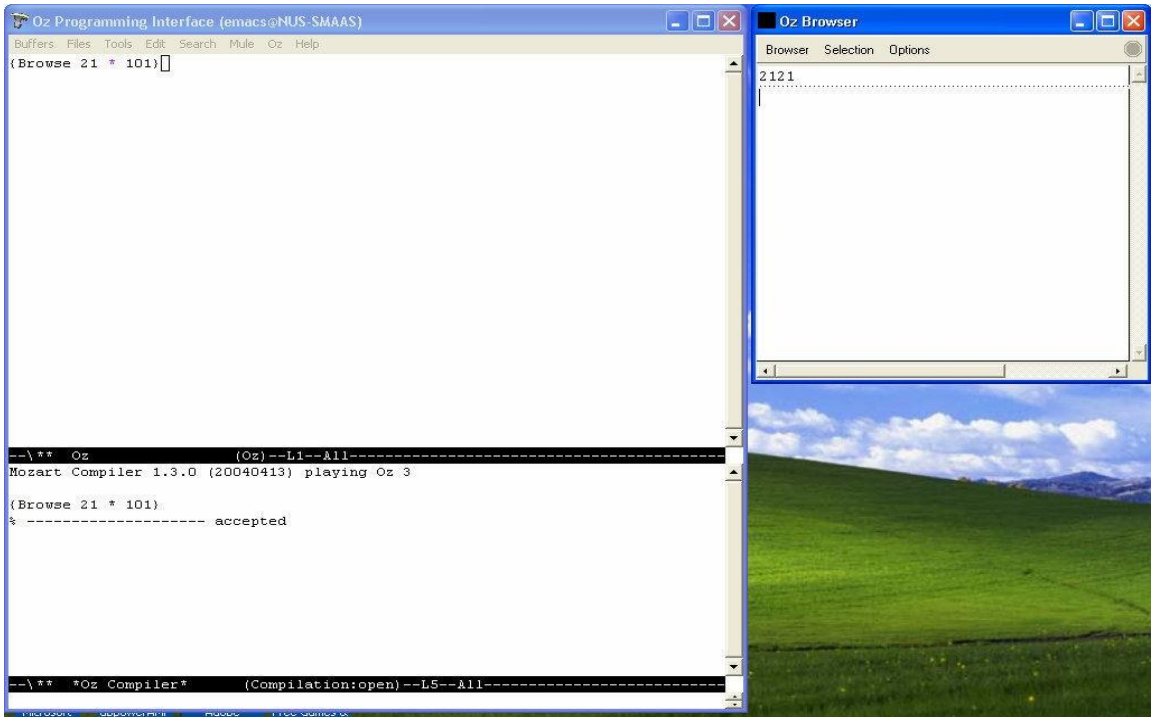
For more details about Mozart commands, you should consult Programming Environment and Tools manual. For more details about emacscommands, you should consult the Emacs on-line tutorial available from the Help menu in the Emacs menu bar or an online tutorial from http://www.lib.uchicago.edu/keith/tcl-course/emacs- tutorial.html.

The Mozart System

- Interactive interface (the declare statement)
  - ‰ Allows introducing program fragments incrementally and execute them
  - ‰ Has a tool (Browser), which allows looking into the store using the procedure Browse
- {Browse 21 * 101} -> by selecting “Oz” panel, “Feed Line” or alternatively “C-. C-l”, this will display in the Browser window the number 2121

Running your first Oz program

The Mozart Interface



Concept of (Single-Assignment) Variable Identifier

```
declare
   X= 21
   X = 22
      % raise an error X =
   21
      % do nothing declare
   X = 22
        % from now on, X will be bound to 22
```

Concept of Oz Variable Type

A variable type is known only after the variable is bound

Examples:

1.
```
   X < 1
   X < 1.0
```

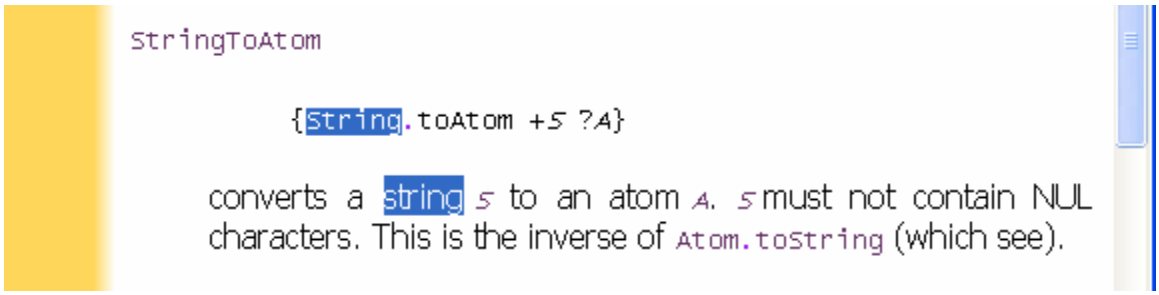
2.
```
declare X Y
   X = "Oz Language"
   Y = 'Oz Language'
   if X == Y
      then {Browse yes}
      else {Browse no}
  end
```

The Mozart Documentation

Please consult the documentation for mozart1 from http://mozart.github.io/documentation/

Concept of Oz Variable Type

```
declare X Y Z
X = "Oz Language"
Y = 'Oz Language'
{String.toAtom X Z}
if Z == Y then {Browse yes}
else {Browse no}
end
```



Try these Functions

```
declare
fun {Minus X}
    ~X
end
{Browse {Minus 15}}
declare
fun {Max X Y}
    if X>Y then X else Y end
    end
declare
X = {Max 22 18}
Y = {Max X 43}
{Browse Y}
```

Exercise 1 (Absolute Value) Write a function Absthat computes the absolute value of a number. This should work for both integers and real numbers.

Try Recursive Function

Recursive function definition

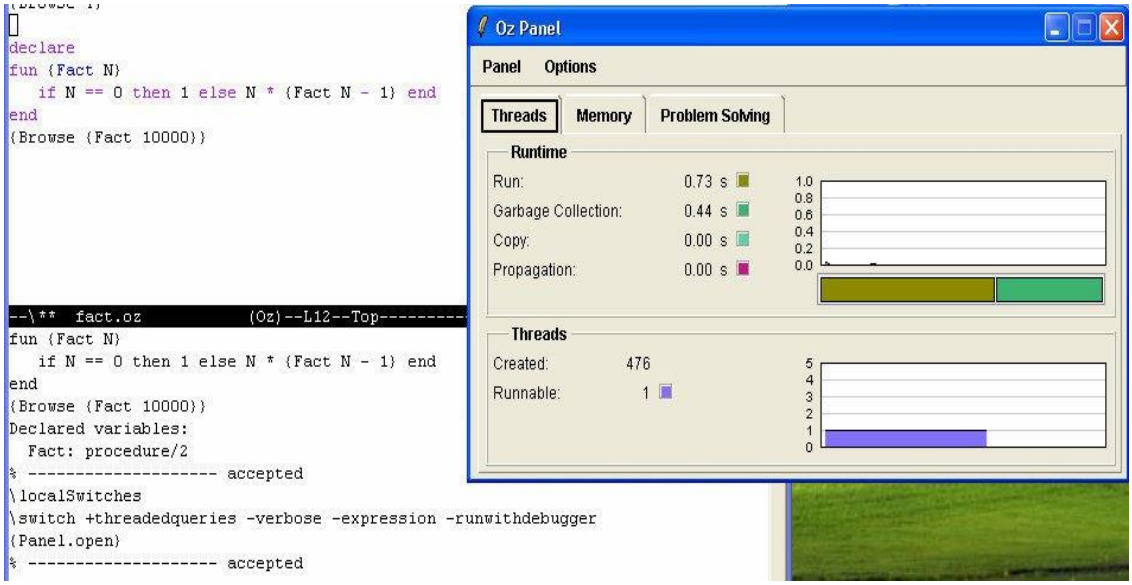
```
fun {Fact N}
    if N == 0 then 1
    else N * {Fact N-1}
    end
end
{Browse {Fact 5}}
```

Try some calls:

- {Fact 5}
- {Fact 100}
- {Fact 10000}

Use the Oz Panel to get an idea how much memory is needed.

Oz Panel



Try Fibonacci Example

The execution time of a program as a function of input size, up to a constant factor, is called the program's **time complexity**.

```
declare
fun {Fibo N}
case N of
        1 then 1
        [] 2 then 1
        [] M then {Fibo (M-1)}  + {Fibo (M-2)}
end
end
{Browse {Fibo 100}}
```

The time complexity of {Fibo N}is proportional to 2^N.

Try Efficient Fibonacci Example

```
declare
fun {FiboTwo N A1 A2}
 case N of
        1 then A1
        [] 2 then A2
        [] M then {FiboTwo (M-1) A2 (A1+A2)}
    end
end
{Browse {FiboTwo 100 1 1}}
```

The time complexity of {Fibo N}is proportional to N.

Exercise 2 (Power) Compute n^m where n is an integer and m is a natural number.
Hint: Use the following inductive definition:

$$n^0 = 1$$
$$n^m = n * n^{m-1}$$

Write a function Powas follows:

```
declare
fun {Pow N M}
  if        ... then
     ...
    else
     ...
   end
end
```

Exercise 3 (Maximum Recursively) Compute the maximum of two natural numbers, knowing that the only allowed test with a conditional is the test whether a number is zero (that is, if N==0 then … else … end).

Hint: Facts about the maximum (n>=0 and m>=0)

- max(n, m)=m, if n=0.
- max(n, m)=n, if m=0.
- max(n, m)=1 + max(n-1, m-1), otherwise.