Front page

NTNU
Faculty of Information Technology
and Electrical Engineering
Department of Computer Science

Endterm examination paper for **TDT4165 Programming Languages**

Academic contact during examination: *Øystein Nytrø*
Phone: *+47 91897606,* Email: *nytroe@ntnu.no*

Examination date: **November 13, 2019**
Examination time (from-to): **17.00-18.00**
Permitted examination support material: **Code E: None**

This course has only an english version of the exam.
This examination has 15 tasks. All tasks have the same weight (1/15).
Wrong answers are not scored negatively.

There is an ungraded text-entry (task 16) at the end of the exam that you can use for comments.

Students will find the examination results in Studentweb after grading has been completed. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer.

1

Interpreting the following statement in Mozart will display...

```
{Browse local Y in local X=[1 2] in X.1 = Y end end}
```

Select one alternative:

- nil
- [1]
- Will display nothing, but raise a unification error
- A syntax error
- 1 ✓

Maximum marks: 1

2

Which of the following statements are true?

Select one alternative:

- Declarative concurrency does not rely on fair scheduling.
- Eager and lazy evaluation are only meaningful with concurrency.
- None of the other alternatives.
- Data-driven concurrency is not declarative.
- Declarative concurrency is not necessarily parallel. ✓

Maximum marks: 1

3

What is the next state of the execution stack after executing the state ('raise B end' ...) (given as a sequence of four tuples *(statement environment)* and current state of the Single Assignment Store).

```
([
   (raise B end              {A->v1, B->v2, C->v3})
   (C=3                      {A->v1, B->v2, C->v3})
   (catch E {Browse E} end   {A->v1, B->v2, C->v3})
   ({Browse A#B#C}           {A->v1, B->v2, C->v3})]   { v1=1, v2=2, v3 })
```

In the below alternative snippets, the elipses '...' are intended to be shorthand for the content matching the given stack state above. (Assume that the identifier E is mapped by the environment to variable v4.)

Select one alternative:

- [(catch E {Browse E} end {A->v1, B->v2, C->v3}) ...] {v1=1, v2=2, v3})
- ([ ] { v1=1, v2=2, v3,v4=v2 })
- [({Browse E}, {A->v1, B->v2, C->v3, E->v4}) ...] {v1=1, v2=2, v3,v4=v2 }) ✓
- ([(raise B end {A->v1, B->v2, C->v3}) (C=3 {A->v1, B->v2, C->v3}) (catch E {Browse E} end {A->v1, B->v2, C->v3}) ...] { v1=1, v2=2, v3 })
- [({Browse A#B#C}, {A->v1, B->v2, C->v3, E->v4})] {v1=1, v2=2, v3,v4=v2 })

Maximum marks: 1

4

The Browse function runs a separate thread when called and shows the current value of the parameter in a Browser-window. Which explanation is most correct when interpreting the following snippet in Mozart?

```
local X in
            thread if X==1 then skip else X=0 end end % thread A
            thread if X==0 then skip else X=1 end end % thread B
            {Browse X}
end
```

Select one alternative:

- '1' or '0' is printed, but that cannot be determined from the code only.
- Nothing is printed and one or both of threads A and B are suspended.
- '0' is printed and one thread suspends.
- '_' is printed and both threads A and B will be suspended. ✓
- '1' is printed and one thread suspends.

Maximum marks: 1

5

Given the Oz code below, and /* insert */ as described, select the correct alternative

```
declare fun lazy {Centiplus N M} N*100 + M end
{Browse /* insert */ {Centiplus 10 20}}
```

Select one alternative:

- When /* insert */ is "1+", the browser will print "1021" ✓
- All of the other alternatives
- When /* insert */ is "{Centiplus 0 0} * " the browser will not show anything because Centiplus threads will suspend.
- When /* insert */ is " ", the browser will not show anything because Centiplus is lazy.
- When /* insert */ is " ", the result will depend upon compiler optimization settings.

Maximum marks: 1

6

Select the correct statement about the application of the Fibonacci-function defined as:

```
fun {Fibonacci N}
   if N<2 then N
   else thread {Fibonacci N-1} end
      + thread {Fibonacci N-2} end
   end
end
```

Select one alternative:

- if N is 2, then the total number of threads created during the execution is 4
- if N is 5, then it is not possible that at some time there will be more than 10 threads running concurrently
- if N is 10, then it is possible that at some time there will be more than 100 threads running concurr ✓ ly
- if N is 10, then it is not possible that at some time there will be less than 10 threads running concurrently.
- If N is 1, no threads will be running.

Maximum marks: 1

7

We want to define an infinite list Integers that contains as many Integers as anyone cares to use. Where should you insert "lazy" below? What is the correct alternative?

```
declare /* 1 */ Integers =
local
   /* 2 */fun /* 3 */ {Generate From}
      From| /* 4 */ {Generate From+1}
   end
in
   {Generate 1}
end
```

Select one alternative:

- /* 2 */
- /* 3 */ ✓
- /* 1 */ and /* 2 */
- None of the other alternatives will realize the intended infinite list of integers.
- /* 1 */

Maximum marks: 1

8

Which statement is true for a declarative program?
Select one alternative:

- It is functional
- All parts of the program execution must be deterministic.
- It will never fail or freeze/suspend.
- It must use logic (immutable) variables as in Oz or 'val'-declared variables in Scala.
- It can use non-declarative constructs or language features. ✓

Maximum marks: 1

9

Which statement about thread safety in Scala is **false**?
Select one alternative:

- Thread safety is not a problem in practice for Scala programs predominantly using logic variables (immutable/val-declarations). ✓
- Thread safety can be achieved by implementing mutual exlusion using 'synchronized' in Scala.
- Thread safe programs will not experience deadlock.
- Mechanisms for ensuring thread safety may reduce the potential for efficient use of parallellism.
- Thread unsafe programs cannot be declarative.

Maximum marks: 1

10

Select the correct alternative completing the sentence "Using 'Either' as the return datatype for a function, ..."
Select one alternative:

- Is a way of implementing if/case sentences when writing Scala in functional style.
- Makes exception-handling more computationally efficient.
- Is better than letting the function throw exceptions, because the function is complete and not partia ✓
- Is used to make the function more flexible by making it polymorphic, ie. able to handle many types.
- All of the other alternatives are correct completions of the sentence.

Maximum marks: 1

11

Select the true statement about Oz:

Select one alternative:

- ○ Exceptions are never deterministic
- ○ Exceptions are never declarative
- ○ Oz can only raise throwable literal values
- ○ Oz allows catch-statements with pattern-matching ✓
- ○ Oz can only raise fully bound values

Maximum marks: 1

12

Select the false statement about Scala:

Select one alternative:

- ○ Atomic execution may prevent race conditions
- ○ Using val-declarations increases declarativeness of programs
- ○ Race conditions is not the result of a deadlock situation.
- ○ Using var-declarations increases declarativeness of programs ✓
- ○ 'x.synchronized' calls enable atomic operations on x.

Maximum marks: 1

13

Given the following relational Oz program

```
local
   fun {Bar X}
      local Y in
         Y = choice c [] d end
         (Y \= X) = true
         X#Y
      end
   end
   fun {Foo}
      choice {Bar a} [] {Bar b} [] {Bar c} end
   end
in
   {Show {SolveAll Foo}}
end
```

If we assume that SolveAll uses depth-first, leftmost-first search. What will be shown?

Select one alternative

- ○ [a#c a#d b#c b#d c#d] ✓
- ○ [c#d b#d b#c a#d a#c]
- ○ None of the other alternatives.
- ○ [a#c a#d b#c b#d c#c c#d]
- ○ [a#c]

Maximum marks: 1

14

Consider the following code written in Scala. Assume that the 'log()'-method will print a string along with the name of the thread calling log.

```
object ThreadsCommunicate extends App {
     var result: String = null
     val t = thread { result = "\nTitle\n" + "=" * 5 }
     // val s = thread { result = "\nTitle\n" + "=" * 2 }
     t.join()
     log(result)
}
```

Which statement is true?

Select one alternative:

- ○ ThreadsCommunicate is an example of nondeterministic concurrency.
- ○ 't.join' ensures that log never prints null. ✓
- ○ If we uncomment the line starting with '// val s', the program may deadlock.
- ○ 't.join' may suspend the thread t.
- ○ If we uncomment the line starting with '// val s', t.join ensures that we can predict log output exactly.

Maximum marks: 1

15

In order to realize relational programming in Oz, the kernel langauge is extended by:

$\vert$ **`choice`** $\langle \mathtt{s} \rangle_1$ `[]` $\cdots$ `[]` $\langle \mathtt{s} \rangle_n$ **`end`**	**Choice**
$\vert$ **`fail`**	**Failure**

Which one of the following statements is false?
Select one alternative:

- Both choice and fail can be implemented by changing SAS-behaviour only.
- Choice-points are generated only for unifications.
- Choice-semantics does not specify a search strategy.
- The 'choice'-statment creates choice-points, so that unifications can be undone.
- The 'fail'-statement is syntactic sugar for raising an exception. ✓

Maximum marks: 1

16

Comment section

This section is not marked or graded, leave comments or clarifications if needed here.

Maximum marks: 0