

CS_411
CRITICAL SYSTEMS
Coursework 1 (5%)

Due: Friday, 1 March 2002, 12:00

Question 1

Solve the following two goals on the theorem prover Agda. You should check using `agda-term-check-buffer`, whether you haven't made a circular definition.

The source for this question can be found in the Linux lab under `~csetzer/examples/ex1.agda`.

```
A :: Type
= Set
```

```
B :: Type
= data
```

```
C :: Type
= data S
```

```
AB :: Type
= sig{a:: A;
      b:: B}
```

```
D :: Type
= sig{q:: (AB -> C) -> (A -> (B-> C));
      r:: (A -> (B->C))-> (AB -> C)}
```

```
{- Try to solve this goal -}
```

```
p:: D
= {! !}
```

```
E :: Type
= sig{q:: ( A -> (B -> C)) -> ((A ->B)-> (A->C));
      r:: ((A ->B)-> (A->C)) -> (A -> (B->C))}
```

```
{- Try to solve this goal-}
```

```
q:: E
= {! !}
```

Question 2

The source for this question can be found in the Linux lab under `~csetzer/examples/ex2.agda`.

(a) Assume the following definitions in Agda:

```
N :: Set
  = data Z | S (n::N)

{- That was the type of natural numbers -}

One :: Set
  = data ast

Vec (A :: Set)
    (n :: N)
  :: Set
  = case n of {
      (Z) -> One ;
      (S n') -> sig{a::A; B:: Vec A n'} ;}

{- Vec A n is A^n if A is any Set -}
```

Here `Vec A n` is the type of vectors of elements of type A of length n .

Define, depending on $n, m :: N$ the type of $n \times m$ -matrices, using the above definitions.

[10 marks]

(b) Assume additionally the following definition of the type `Bool` of Booleans:

```
Bool :: Set
  = data false | true
```

Use this definition in order to define the type of digital components, where a digital component consists of natural numbers n, m and a function from Bool^n to Bool^m .

[10 marks]

Question 3

Solve the following two goals on the theorem prover Agda. You should check using `agda-term-check-buffer`, whether you haven't made a circular definition.

The source for this question can be found in the Linux lab under `~csetzer/examples/ex3.agda`.

package Ex (X,Y,Z:: Type) where

S :: Type
= sig{ fst:: X; snd:: X-> Y}

T :: Type
= S -> Y

P :: T
= \ (h::S) -> h.snd h.fst

Q :: (((X -> Y)-> Z)-> Z) -> X -> (Y -> Z)-> Z
= {! !}

R :: ((X ->Z)-> Z) -> (((X->Y)->Z)->Z) -> (Y->Z)->Z
= {! !}

[15 marks]