1) Construir a especificação;
2) Realizar testes;
3) Comentar sobre as correspondências entre parâmetros;
4) Construir uma especificação real a partir de 6.4.3 e testar;

5) Data de entrega 16/04

6.4.3.TWO-DISJOINT-UNIONS { ARRAYS{ X :: PAR }, DOUBLES { Y :: PAR, Z :: PAR }}

Obs: No 3, usar tags.

1) Construir a especificação;

```
(fth PAR is
sort Component .
endfth)

(fmod LISTS { X :: PAR } is
protecting NAT .
sort List{X} . *** sort estruturado
op empty-list : -> List{X} [ctor] .
op cons : X$Component List{X} ->List{X} [ctor] .
op head`of_ : ~> X$Component .
op tail`of_ : List{X} -> List{X} .
op length`of_ : List{X} -> Nat .
op is`empty-list_ : List{X} -> Bool .
var c c' : X$Component .
var l : List{X} .
eq head of cons (c, l) = c .
eq tail of cons (c, l) = l .
eq tail of empty-list = empty-list .
eq length of empty-list = 0 .
eq length of cons(c, l) = 1 + length of l .
eq is empty-list empty-list = true .
eq is empty-list cons (c, l) = false .
endfm)


fmod ARRAYS { X::PAR} is
protecting INT .
sort Array .
op unit-array : X$Component -> Array [ctor] .
op _ abutted to _ : Array Array -> Array [ assoc ctor] .
```

```
op component _ of _ : Nat Array ~> X$Component .
op size of _ : Array -> Nat .
var a a' a'' : Array .
var c : X$Component .
var i : Int .
eq component 0 of unit-array(c) = c .
eq component 0 of unit-array(c) abutted to a = c .
ceq component i of unit-array(c) abutted to a =
component (i-1) of a
 if i > 0 .
eq size of unit-array(c) = 1 .
eq size of unit-array(c) abutted to a = 1 +
 (size of a) .
endfm

(view Par-as-Arrays{X1::PAR}
from PAR to ARRAYS{X1} is
sort Component to Arrays{X1} .
endv)

(view Par-as-Double { X1 :: PAR , X2 :: PAR }
from PAR to DOUBLES { X1 , X2 } is
sort Component to Double { X1 , X2 } .
endv)

fmod TWO-DISJOINT-UNION{X1:: PAR, X2:: PAR} is
sort TwoDisjointUnion .
op tag1_ : X1$Component -> TwoDisjointUnion [ctor].
op tag2_ : X2$Component -> TwoDisjointUnion [ctor].
endfm

(fmod COMPOSITION { X :: PAR , Y :: PAR , Z :: PAR } is
protecting TWO-DISJOINT-UNION { Par-as-Arrays { X },
 Par-as-Double { Y , Z } } .
endfm)
```

2) Testar as especificações

Não consegui testar adequadamente pois fiquei um pouco confuso sobre como passar os inputs. Tentei algumas coisas e me deparei com erros no Maude.

3) Comentar sobre as correspondências entre parâmetros

A correspondência entre parâmetros foi realizada através de visões, mecanismo que mapeia uma teoria a uma especificação parametrizada. Para poder fazer o mapeamento conforme proposto no exercicio foi adicionada uma camada composition na qual estendemos as operações de TwoDisjointUnion e criamos um mapeamento conforme a entrada proposta no exercício.

4) Construir uma especificação real a partir de 6.4.3 e testar;

```
(fmod REAL-SPEC is
protecting COMPOSITION {Par-as-Arrays, Par-as-Double, Par-as-Double} .
endfm)
```