QCM : MA121 HA PREUVE

12 janvier 2023

Durée : 15 minutes.

Aucun document n'est autorisé. L'usage de la calculatrice est interdit. Les questions suivies de ♣ peuvent admettre plusieurs bonnes réponses. Pour chacune des autres questions, il y a 1 et 1 seule bonne réponse. Une bonne réponse rapporte des points mais une mauvaise en fait perdre. Une non-réponse (aucune case cochée) ne rapporte rien mais ne coûte rien.

☐0	☐0	☐0	☐0	☐0	☐0	☐0	☐0
☐1	☐1	☐1	☐1	☐1	☐1	☐1	☐1
☐2	☐2	☐2	☐2	☐2	☐2	☐2	☐2
☐3	☐3	☐3	☐3	☐3	☐3	☐3	☐3
☐4	☐4	☐4	☐4	☐4	☐4	☐4	☐4
☐5	☐5	☐5	☐5	☐5	☐5	☐5	☐5
☐6	☐6	☐6	☐6	☐6	☐6	☐6	☐6
☐7	☐7	☐7	☐7	☐7	☐7	☐7	☐7
☐8	☐8	☐8	☐8	☐8	☐8	☐8	☐8
☐9	☐9	☐9	☐9	☐9	☐9	☐9	☐9

⟵ codez votre numéro d'étudiant ci-contre, et inscrivez votre nom et prénom ci-dessous.

Nom et prénom :

....

Question 1 Comment démontrer une assertion de la forme $P \Longrightarrow Q$?

☐ On suppose Q, puis on démontre P
☐ On suppose P, puis on démontre Q
☐ On démontre P ou Q
☐ On démontre P, puis on démontre Q
☐ On démontre P, puis on suppose Q
☐ On démontre Q
☐ On démontre P
☐ On démontre P et Q
☐ On démontre Q, puis on suppose P

Question 2
Dans L∃∀N , comment introduire le connecteur → dans `h: (P → Q)` ?

☐ `and.intro hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `implies.intro h`
☐ `h hQ` où `hQ` est une preuve de `Q`
☐ `implies.elim h`
☐ `h hP` où `hP` est une preuve de `P`
☐ `assume hP:P,`
☐ `implies.elim h hP` où `hP` est une preuve de `P`
☐ `hP → hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `assume hQ:Q,`

Question 3 Dans L∃∀N , comment introduire le connecteur ∀ dans `h: (∀x:E, P x)` ?

☐ `forall.elim h`
☐ `h x`
☐ `forall.intro x`
☐ `soit x:E,`
☐ `assume x:E,`
☐ `let x:E,`
☐ `let x∈E,`
☐ `forall.intro h x`
☐ `∀ x:E,`
☐ `∀ x∈E,`

Question 4
Dans L∃∀N , comment éliminer le connecteur → dans `h: (P → Q)` ?

☐ `assume hP:P,`
☐ `or.elim h`
☐ `or.elim hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `hP → hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `implies.elim h hP` où `hP` est une preuve de `P`
☐ `implies.intro h`
☐ `h hP` où `hP` est une preuve de `P`
☐ `h hQ` où `hQ` est une preuve de `Q`
☐ `implies.elim h`
☐ `assume hQ:Q,`

Question 5 ♣ Dans L∃∀N , comment éliminer le connecteur ∧ dans `h: (P ∧ Q)` ?

☐ `and.intro hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `or.intro hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `and.elim hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `and.intro h`
☐ `or.intro h`
☐ `and.inl hP` où `hP` est une preuve de `P`
☐ `h.right`
☐ `and.elim h`
☐ `h.left`
☐ `or.elim h`
☐ `or.inr hQ` où `hQ` est une preuve de `Q`
☐ `or.elim hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `and.inr hQ` où `hQ` est une preuve de `Q`
☐ `h hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `or.inl hP` où `hP` est une preuve de `P`

Question 6 ♣ Dans L∃∀N , comment introduire le connecteur ∨ dans `h: (P ∨ Q)` ?

☐ `and.intro hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `and.inr hQ` où `hQ` est une preuve de `Q`
☐ `or.inr hQ` où `hQ` est une preuve de `Q`
☐ `h hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `and.inl hP` où `hP` est une preuve de `P`
☐ `or.inl hP` où `hP` est une preuve de `P`
☐ `or.elim hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `or.intro hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `or.elim h`
☐ `and.elim h`
☐ `or.intro h`
☐ `and.elim hP hQ` où `hP` et `hQ` sont des preuves de `P` et `Q`
☐ `h.left`
☐ `h.right`
☐ `and.intro h`

Question 7

Traduisez le code source en lean ci-dessous en un texte mathématique (en français) cohérent (définitions, théorème : énoncé et preuve, etc).

```
variables {E F G:Type}

definition injective  (f: E → F) : Prop  := ∀ (u:E), ∀ (v:E), f u = f v  →  u = v
definition surjective (f: E → F) : Prop  := ∀ (y:F), ∃ (x:E), y = f x

theorem interessant : ∀  (u: E → F) (v: F → G), surjective (v ∘ u) → (injective v) → (surjective u) :=
assume  (u: E → F) (v: F → G),
  assume h1 : surjective (v ∘ u),
    assume h2: injective v,
      assume (y:F),
        let z:= v y in
         exists.elim (h1 (z:G))
         (
           assume (x:E) (h3: z = (v ∘ u) x),
           have h4: v y = v (u x), from h3,
           exists.intro x (
             show y = u x, from h2 y (u x) h4
           )
         )
```

□0 □1 □2 □3 □4 □5 □6 □7 □8 □9 □10 □11 □12 NE RIEN COCHER