15 Modularity

1. (a)
```
scheme
  PSTACK1(E : ELEMENT) =
    class
      type Stack == empty | push(top : E.Elem, pop : Stack)

      value
        is_empty : Stack → Bool
        is_empty(st) ≡ st = empty
    end

scheme
  ELEMENT = class type Elem end
```

   (b)
```
scheme
  INT_STACK = PSTACK1(INTEGER)

object
  INTEGER : class type Elem = Int end
```

   (c)
```
scheme
  PSTACK2(E : ELEMENT) =
    class
      type Stack = E.Elem*

      value
        empty : Stack = ⟨⟩,

        push : E.Elem × Stack → Stack
        push(e, st) ≡ ⟨e⟩ ⌢ st,

        top : Stack ~→ E.Elem
        top(st) ≡ hd st pre st ≠ ⟨⟩,

        pop : Stack ~→ Stack
        pop(st) ≡ tl st pre st ≠ ⟨⟩,

        is_empty : Stack → Bool
        is_empty(st) ≡ st = ⟨⟩
    end
```

2.
```
scheme
  STACK_STACK(E : ELEMENT) =
    extend class object S : PSTACK1(E) end with
      use Stack_of_stacks for Stack in
        PSTACK1(S{Stack for Elem})
```

```
scheme
  STACK_STACK(E : ELEMENT) =
    class
      object
        S :
          class
            type Stack == empty | push(top : E.Elem, pop : Stack)

            value
              is_empty : Stack → Bool
              is_empty(st) ≡ st = empty
          end

      type Stack_of_stacks == empty | push(top : S.Stack, pop : Stack_of_stacks)

      value
        is_empty : Stack_of_stacks → Bool
        is_empty(st) ≡ st = empty
    end
```