```
─────────────────────────── MODULE building ───────────────────────────
  Sample solution for first TLA+ exercise Extended to include permissions

CONSTANT
    People,       we're dealing with people here
                  this is the set of all people
    Buildings   Set of all buildings

VARIABLE
    register,     Set of registered users
    permission,   Function giving permissions
    location      Function giving location

outside ≜ CHOOSE x : x ∉ Buildings  outside is not in a Building
  To check the above, TLC throws an error "TLC attempted to evaluate an unbounded CHOOSE ."
  The solution to this is to use the "Additional Spec Options" →  "definition override" option If
  you create a new model, this gets filled in for you
TypeOK ≜   type invarient
    ∧  register ⊆ People        Everyone on the register is a person
    ∧  permission ∈ [register → SUBSET Buildings]
    ∧  location ∈ [register → Buildings ∪ {outside}]
    ∧  outside ∉ Buildings

────────────────────────────────────────────────────────────────────────

Init ≜
   ∧ register     = {}        Initially no-one is registered
   ∧ permission = [x ∈ {} ↦ {}]            no-one has permissions
   ∧ location     = [x ∈ {} ↦ outside]         no-one is anywhere


Register(p) ≜
    ∧   p ∈ People \ register            p is a person and not registered
    ∧ register′ = register ∪ {p}         add p to register
    ∧ permission′ = [x ∈ DOMAIN permission ∪ {p} ↦
                         IF x ∈ DOMAIN permission
                          THEN permission[x]
                          ELSE {}]
    ∧ location′     = [x ∈ DOMAIN location ∪ {p} ↦
                         IF x ∈ DOMAIN location
                          THEN location[x]
                          ELSE outside]

DeRegister(p) ≜   Unregister a person
    ∧ p ∈ register                                    person is registered
    ∧ location[p] = outside                           person is outside
    ∧ register′ = register \ {p}
    ∧ location′    = [x ∈ DOMAIN location \ {p} ↦          remove p from domain of location
                        location[x]]                       preserve other locations
    ∧ permission′ = [x ∈ DOMAIN permission \ {p} ↦         remove p from domain of permissions
                        permission[x]]                     preserve other permissions

AddPermission(p, b) ≜   add permission for person p for building b
    ∧ p ∈ register                      p is registered
    ∧ p ∈ DOMAIN permission             p has permissions
    ∧ permission′ = [permission EXCEPT ![p] = @ ∪ {b}]  add b to permissions for p
    ∧ UNCHANGED location
    ∧ UNCHANGED register

RevokePermission(p, b) ≜
    ∧ p ∈ register
    ∧ p ∈ DOMAIN permission
    ∧ permission′ = [permission EXCEPT ![p] = @ \ {b}]  remove b from permissions for p
    ∧ UNCHANGED location
    ∧ UNCHANGED register

Enter(p, b) ≜
    ∧ p ∈ register          p is registered
    ∧ b ∈ permission[p]     p has permission to enter b
    ∧ location′ = [location EXCEPT ![p] = b]   update p’s location
    ∧ UNCHANGED register       does not change who is registered
    ∧ UNCHANGED permission     does not change permissions

Leave(p, b) ≜
    ∧ p ∈ register            p is registered
    ∧ location[p] = b         p is in the building
    ∧ location′ = [location EXCEPT ![p] = outside]  update p’s location now outside
    ∧ UNCHANGED register         don’t change register
    ∧ UNCHANGED permission       don’t change permissions


Next ≜
    ∃ p ∈ People, b ∈ Buildings :  There is a person and a building that
        ∨ Register(p)                   the person can be registered
        ∨ DeRegister(p)                 the person can be un-registered
        ∨ AddPermission(p, b)           the person can have a permission
        ∨ RevokePermission(p, b)        the person can have permission
        ∨ Enter(p, b)                 the person can enter the building
        ∨ Leave(p, b)               the person can leave the building
```

* Modification History
* Last modified *Wed Feb* 10 01:46:58 *GMT* 2021 by *alunm*
* Last modified *Tue Feb* 09 21:24:32 *GMT* 2021 by *alunm*

\ * Last modified *Wed Oct* 02 10:31:48 *BST* 2019 by *alun*
\ * Last modified *Tue Sep* 10 12:27:57 *BST* 2019 by *cgam*1
\ * Created *Mon Sep* 24 11:53:39 *BST* 2018 by *cgam*1