galois

Exercise 1:

```
// The puzzle goes like this:
// You've got x coins that add up to s cents - what are they?
// Use currying - for this example #coins is 30, total sum is 109 cents

coinPuzzle : [10] -> [10] -> [10] -> [10] -> [10] -> [10] -> Bit
coinPuzzle n s a b c d = (coinCount a b c d n) /\ (coinSum a b c d s)

coinSum : [10] -> [10] -> [10] -> [10] -> [10] -> Bit
coinSum   a b c d s = (a + 5*b + 10*c + 25*d) == s

coinCount : [10] -> [10] -> [10] -> [10] -> [10] -> Bit
coinCount a b c d x = (((a + b + c + d) == x) /\ // the coin count adds up
                       (a <= x /\ b <= x /\ c <= x /\ d <= x))

check : [10] -> [10] -> [10] -> [10] -> [10]
check a b c d = a*1 + b*5  + c*10 + d*25
```

```
// run it like this:
// Main> :set base=10
// Main> :set satNum = all
// Main> :sat coinPuzzle 30 109
// Satisfiable
// coinPuzzle 30 109 19 7 3 1 = True
// coinPuzzle 30 109 24 1 3 2 = True
// coinPuzzle 30 109 14 13 3 0 = True
// coinPuzzle 30 109 19 4 7 0 = True
// Models found: 4
// (Total Elapsed Time: 0.055s, using "Z3")
```

Exercise 2:

variables:

for all `i` from `1` to `9`:

declare `bi` with the interpretation that `bi` is true iff the Bitcoin is in room `i`

declare `ti` with the interpretation that `ti` is true iff there is a Tiger in room `i`

declare `si` with the interpretation that `si` is true iff the sign on the door of room `i` is true

constraints:

```
// True iff the Bitcoin is in exactly one room
Bitcoin_in_one_room b1 b2 b3 b4 b5 b6 b7 b8 b9 =
  ((~b1) \/ (~b2)) /\ ((~b1) \/ (~b3)) /\ ((~b1) \/ (~b4)) /\ ((~b1) \/ (~b5)) /\ ((~b1) \/ (~b6)) /\
  ((~b1) \/ (~b7)) /\ ((~b1) \/ (~b8)) /\ ((~b1) \/ (~b9)) /\ ((~b2) \/ (~b3)) /\ ((~b2) \/ (~b4)) /\
  ((~b2) \/ (~b5)) /\ ((~b2) \/ (~b6)) /\ ((~b2) \/ (~b7)) /\ ((~b2) \/ (~b8)) /\ ((~b2) \/ (~b9)) /\
  ((~b3) \/ (~b4)) /\ ((~b3) \/ (~b5)) /\ ((~b3) \/ (~b6)) /\ ((~b3) \/ (~b7)) /\ ((~b3) \/ (~b8)) /\
  ((~b3) \/ (~b9)) /\ ((~b4) \/ (~b5)) /\ ((~b4) \/ (~b6)) /\ ((~b4) \/ (~b7)) /\ ((~b4) \/ (~b8)) /\
  ((~b4) \/ (~b9)) /\ ((~b5) \/ (~b6)) /\ ((~b5) \/ (~b7)) /\ ((~b5) \/ (~b8)) /\ ((~b5) \/ (~b9)) /\
  ((~b6) \/ (~b7)) /\ ((~b6) \/ (~b8)) /\ ((~b6) \/ (~b9)) /\ ((~b7) \/ (~b8)) /\ ((~b7) \/ (~b9)) /\
  ((~b8) \/ (~b9)) /\ (b1 \/ b2 \/ b3 \/ b4 \/ b5 \/ b6 \/ b7 \/ b8 \/ b9)
```

```
// True iff the Bitcoin is not in the same room as a Tiger
Bitcoin_not_in_tiger_room b1 b2 b3 b4 b5 b6 b7 b8 b9 t1 t2 t3 t4 t5 t6 t7 t8 t9 =
  ((~b1) \/ (~t1)) /\ ((~b2) \/ (~t2)) /\ ((~b3) \/ (~t3)) /\ ((~b4) \/ (~t4)) /\ ((~b5) \/ (~t5)) /\
  ((~b6) \/ (~t6)) /\ ((~b7) \/ (~t7)) /\ ((~b8) \/ (~t8)) /\ ((~b9) \/ (~t9))

assign_values_to_signs b1 b2 b3 b4 b5 b6 b7 b8 b9 t1 t2 t3 t4 t5 t6 t7 t8 t9
                       s1 s2 s3 s4 s5 s6 s7 s8 s9 =
  (s1 == (b1 \/ b3 \/ b5 \/ b7 \/ b9)) /\   /* s1 iff bitcoin in odd number room         */
  (s2 == ((~t2) /\ (~b2))) /\               /* s2 iff room 2 empty                      */
  (s3 == (s5 \/ (~s7))) /\                  /* s3 iff sign 5 is true or sign 7 is false */
  (s4 == (~s1)) /\                          /* s4 iff sign 1 is false                   */
  (s5 == ((~s2) \/ s4)) /\                  /* s5 iff sign 2 is false or sign 4 is true */
  (s6 == (~s3)) /\                          /* s6 iff sign 3 is false                   */
  (s7 == (~b1)) /\                          /* s7 iff bitcoin not in room 1             */
  (s8 == (t8 /\ (~t9) /\ (~b9))) /\         /* s8 iff room 8 has tiger & room 9 is empty */
  (s9 == (t9 /\ (~s6))) /\                  /* s9 iff room 9 has tiger & sign 8 is false */
  (~t1 \/ ~s1) /\                           /* if tiger in room 1 sign in room 1 false   */
  (~t2 \/ ~s2) /\                           /* if tiger in room 2 sign in room 2 false   */
  (~t3 \/ ~s3) /\                           /* if tiger in room 3 sign in room 3 false   */
  (~t4 \/ ~s4) /\                           /* if tiger in room 4 sign in room 4 false   */
  (~t5 \/ ~s5) /\                           /* if tiger in room 5 sign in room 5 false   */
  (~t6 \/ ~s6) /\                           /* if tiger in room 6 sign in room 6 false   */
  (~t7 \/ ~s7) /\                           /* if tiger in room 7 sign in room 7 false   */
  (~t8 \/ ~s8) /\                           /* if tiger in room 8 sign in room 8 false   */
  (~t9 \/ ~s9) /\                           /* if tiger in room 9 sign in room 9 false   */
  (~b1 \/ s1) /\                            /* if bitcoin in room 1 sign 1 is true       */
  (~b2 \/ s2) /\                            /* if bitcoin in room 2 sign 2 is true       */
  (~b3 \/ s3) /\                            /* if bitcoin in room 3 sign 3 is true       */
  (~b4 \/ s4) /\                            /* if bitcoin in room 4 sign 4 is true       */
  (~b5 \/ s5) /\                            /* if bitcoin in room 5 sign 5 is true       */
  (~b6 \/ s6) /\                            /* if bitcoin in room 6 sign 6 is true       */
  (~b7 \/ s7) /\                            /* if bitcoin in room 7 sign 7 is true       */
  (~b8 \/ s8) /\                            /* if bitcoin in room 8 sign 8 is true       */
  (~b9 \/ s9)                               /* if bitcoin in room 9 sign 9 is true       */

// room is true or false - one of b1, b2, b3, b4, b5, b6, b7
// so this function returns true or false - true if the bitcoin is in the input room
// and false otherwise
Bitcoin_in_room room = room

property Bitcoin_and_tiger b1 b2 b3 b4 b5 b6 b7 b8 b9
                           t1 t2 t3 t4 t5 t6 t7 t8 t9
                           s1 s2 s3 s4 s5 s6 s7 s8 s9 =
   ~((Bitcoin_in_one_room b1 b2 b3 b4 b5 b6 b7 b8 b9) /\
     (Bitcoin_not_in_tiger_room b1 b2 b3 b4 b5 b6 b7 b8 b9 t1 t2 t3 t4 t5 t6 t7 t8 t9) /\
     (assign_values_to_signs b1 b2 b3 b4 b5 b6 b7 b8 b9
                             t1 t2 t3 t4 t5 t6 t7 t8 t9
                             s1 s2 s3 s4 s5 s6 s7 s8 s9) /\
     t8 \/ b8)            /* Room 8 is occupied                    */
 \/ (Bitcoin_in_room b7)  /* Proves that Bitcoin must be in Room 7 */

// Run it like this:
// Main> :prove Bitcoin_and_tiger
// Q.E.D.
// (Total Elapsed Time: 0.023s, using "Z3")
```