Exercise 1

Formula 1-1

F_1 = {a = c, d = f(f(c)), f(c) = f(f(f(b))), f(b) = a, d ≠ f(c)}
[Partition F_1]
{{a}, {c}, {d}, {f(f(c))}, {f(c)}, {f(f(f(b)))}, {f(b)}, {f(c)}}
[Merge congruence classes]
{{a, c, f(b)}, {d, f(f(c))}, {f(c), f(f(f(b)))}}
[Propagate (c = f(b)) => (f(c) = f(f(b)))]
{{a, c, f(b)}, {d, f(f(c))}, {f(c), f(f(f(b))), f(f(b))}}
[Propagate (f(c) = f(f(b))) => (f(f(c)) = f(f(f(b))))]
{{a, c, f(b)}, {d, f(f(c)), f(c), f(f(f(b))), f(f(b))}}
[d and f(c) is in the same congruence class]
UNSATISFIABLE

Formula 1-2

F_2 = {a = b, c = f(d), f(b) = g(a), d = c, g(b) = d, g(c) ≠ f(f(a))}
[Partition F_2]
{{a}, {b}, {c}, {f(d)}, {f(b)}, {g(a)}, {d}, {g(b)}, {g(c),{f(f(a))}}
[Merge congruence classes]
{{a, b}, {c, f(d), d, g(b)}, {f(b), g(a)}, {g(c)}, {f(f(a))}}
[Propagate (a = b) => (g(a) = g(b))]
{{a, b}, {c, f(d), d, g(b), g(a), f(b)}, {g(c)}, {f(f(a))}}
[Propagate (a = b) => (f(f(a)) = f(f(b)))]
{{a, b}, {c, f(d), d, g(b), g(a), f(b)}, {g(c)}, [f(f(a)), f(f(b))}}
[Propagate (d = f(b)) => (f(d) = f(f(b)))]
{{a, b}, {c, f(d), d, g(b), g(a), f(b), f(f(a)), f(f(b))}, {g(c)}}
[Every equality from F_2 is satisfied within the congruence classes]
[Inequal clauses are in separate classes, theorem is satisfied]
SATISFIABLE

Exercise 3

Formula 1-1

Encoding:

```
; Literals
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)
(declare-fun d () Int)

; Functions
(declare-fun f (Int) Int)

; Assertions
(assert (= a c))
(assert (= d (f (f c))))
(assert (= (f c) (f (f (f b)))))
(assert (= (f b) a))
(assert (not (= (f b) a)))

; Results
(check-sat)
```

Result:

```
unsat
```

Formula 1-2

Encoding:

```
; Literals
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)
(declare-fun d () Int)

; Functions
(declare-fun f (Int) Int)
(declare-fun g (Int) Int)
```

```
; Assertions
(assert (= a b))
(assert (= c (f d)))
(assert (= (f b) (g a)))
(assert (= d c))
(assert (= (g b) d))
(assert (not (= (g c) (f (f a)))))

; Results
(check-sat)
(get-model)
```

Result:

```
sat
(model
     (define-fun b () Int 0)
     (define-fun c () Int 1)
     (define-fun d () Int 1)
     (define-fun a () Int 0)
     (define-fun f ((x!0 Int)) Int
          (ite (= x!0 1) 1
          (ite (= x!0 0) 1 1)))
     (define-fun g ((x!0 Int)) Int
          (ite (= x!0 0) 1
          (ite (= x!0 1) 2 1)))
)
```

Formula 2

Encoding:

```
; Literals
(declare-fun a () Int)
(declare-fun b () Int)
(declare-fun c () Int)

; Functions
(declare-fun f (Int) Int)
```

```
; Assertions
(assert (>= b 0))
(assert (= (f a) c))
(assert (>= (f a) a))
(assert (>= (- a b) c))
(assert (>= (f c) 0))

; Results
(check-sat)
(get-model)
```

Result:

```
sat
(model
     (define-fun b () Int 0)
     (define-fun a () Int 0)
     (define-fun c () Int 0)
     (define-fun f ((x!0 Int)) Int
          (ite (= x!0 0) 0 0))
)
```

Exercise 4

Sanitizer 1 - Equality

Encoding:

```
; Literals
(declare-fun a () String)

; Functions
(define-fun sanitize1 ((x!0 String)) String
  (ite (str.contains x!0 "<")
    (str.substr x!0 0 (str.indexof x!0 "<"))
    a
  )
)

; Assertions
(assert (= (sanitize1 a) "<script>"))
```

```
; Results
(check-sat)
(get-model)
```

Result:

```
unsat
Z3(17, 10): ERROR: model is not available
```

Sanitizer 1 – Contains

Encoding:

```
; Literals
(declare-fun a () String)

; Functions
(define-fun sanitize1 ((x!0 String)) String
  (ite (str.contains x!0 "<")
    (str.substr x!0 0 (str.indexof x!0 "<"))
    a
  )
)

; Assertions
(assert (str.contains (sanitize1 a) "<script>"))

; Results
(check-sat)
(get-model)
```

Result:

```
sat
(model
 (define-fun a () String
   "\x00<script>\x00<")
)
```

Sanitizer 2 – Equality

Encoding:

```
     ; Literals
     (declare-fun a () String)

     ; Functions
     (define-fun sanitize2 ((x!0 String)) String
       (ite (str.contains x!0 "<")
         (str.replace x!0 "<" "")
         a
       )
)
     )

     ; Assertions
     (assert (= (sanitize2 a) "<script>"))

     ; Results
     (check-sat)
     (get-model)
```

Result:

```
     sat
     (model
       (define-fun a () String
         "<<script>")
     )
```

Sanitizer 2 – Contains

Encoding:

```
; Literals
(declare-fun a () String)

; Functions
(define-fun sanitize2 ((x!0 String)) String
  (ite (str.contains x!0 "<")
    (str.replace x!0 "<" "")
    a
  )
)

; Assertions
(assert (str.contains (sanitize2 a) "<script>"))

; Results
(check-sat)
(get-model)
```

Result:

```
sat
(model
     (define-fun a () String
          "<\x00<script>")
)
```