**BAHRIA UNIVERSITY,**

**(Karachi Campus)**

# Department of Software Engineering



# **ASSIGNMENT # 03 – Fall 2023**

## 

COURSE TITLE: **Formal Methods in Software Engineering** COURSE CODE: **SEN - 323**

CLASS: **BSE-5 (A, B)** SHIFT: **Morning**

INSTRUCTOR: **ENGR. AMMARAH KHALID** DATE: **21st Dec 2023**

MAX. MARKS: **10**

Name: **Faiq Bin Sabir**

Reg.no: **79317(02-131212-067)**

**ASSIGNMENT # 03 (CLO 3)**

***This is a Complex Engineering Problem Assignment. (Mapped on attributes A & C)***

**TASK # 01:**

**A)** Model the transition system of traffic lights given in Figure # 01 in NuSMV

**B)** Print initial state and all reachable states of above transition system using NuSMV.

**C)** Write the safety property of traffic light system formally and informally.

**SOLUTION:**

**PART-A**

MODULE main

VAR

state : {initial, red, warn\_red, green, warn\_green};

ASSIGN

init(state) := initial;

next(state) :=case

state = initial : red;

state = red : warn\_red;

state = warn\_red : green;

state = green : warn\_green;

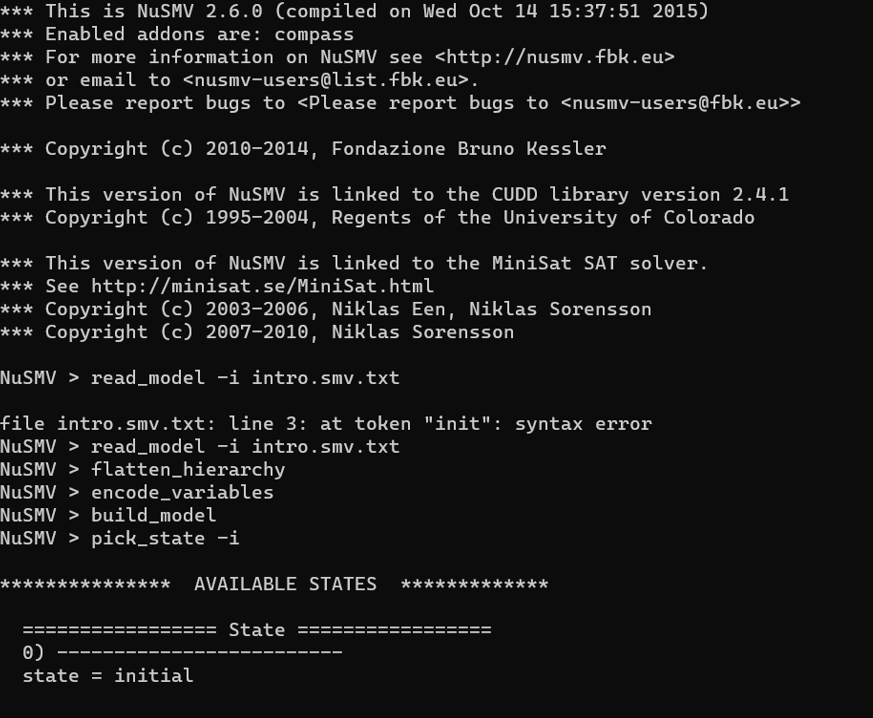
state = warn\_green : red;

esac;

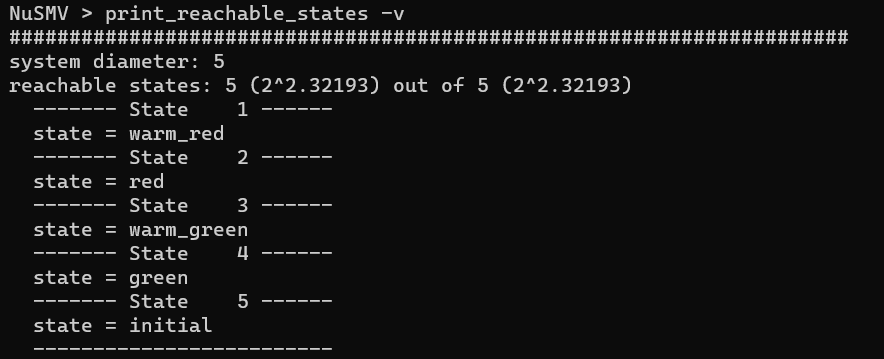
**PART-B**

**INTIAL STATES:**

****

****

**REACHABLE STATES:**

****

**PART-C**

* **Informally**:

A traffic light system should never show green in all directions at the same time to avoid collisions.

* **Formally**:

AG ¬ (light = green ∧ light = green)

* **Informally**:

“A single traffic light system on an entire road should never show green and red at the same time.”

* **Formally**:

AG ¬ (light = green ∧ light=red)

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

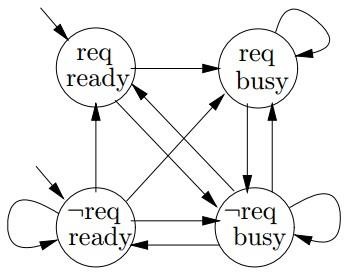
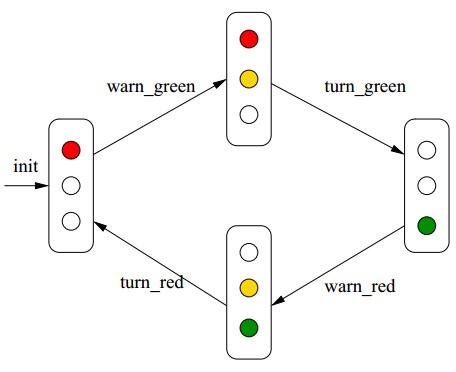
**TASK # 02:**

1. Model the transition system given in Figure # 02 in NuSMV.
2. Print initial state and all reachable states of above transition system using NuSMV
3. Simulate this model for 5 steps
4. Identify whether the given LTL specification are true or false for the model (**Figure. 2**) using

NuSMV

* 1. G(request -> F status=busy)
  2. X(request = FALSE)

*Figure 1: Traffic Light*



*Figure*



*2*



*:*



*Transition System*



**PART-A**

MODULE main

VAR

req: boolean;

status: {ready, busy};

ASSIGN

init(req) := FALSE;

init(status) := ready;

next(req) := case

status = ready: TRUE;

status = busy: FALSE;

esac;

next(status) := case

req & status = ready: busy;

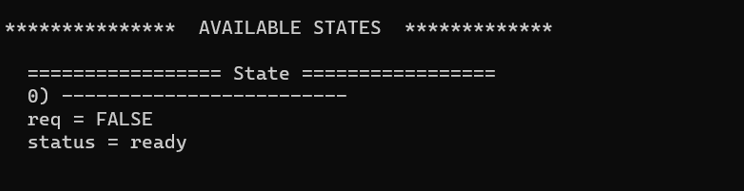
!req & status = busy: ready;

TRUE: status;

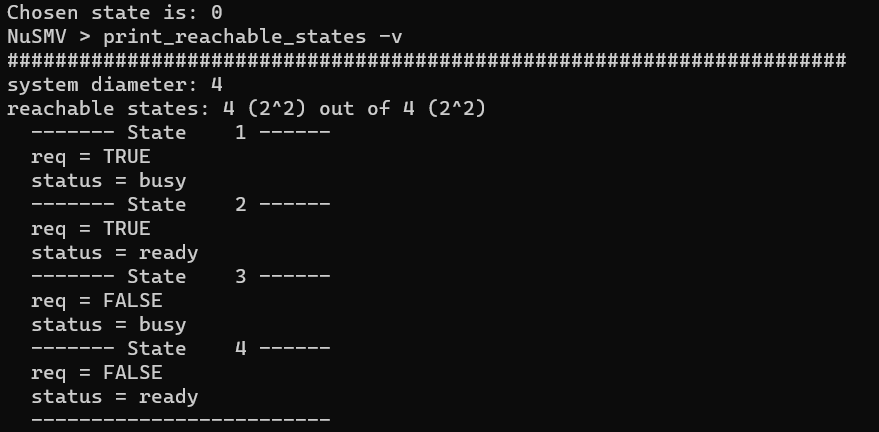
esac;

**PART-B**

**INTIAL STATES:**



**REACHABLE STATES:**



**PART-C**

A screenshot of a computer program

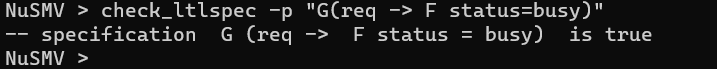
Description automatically generated

A screen shot of a computer

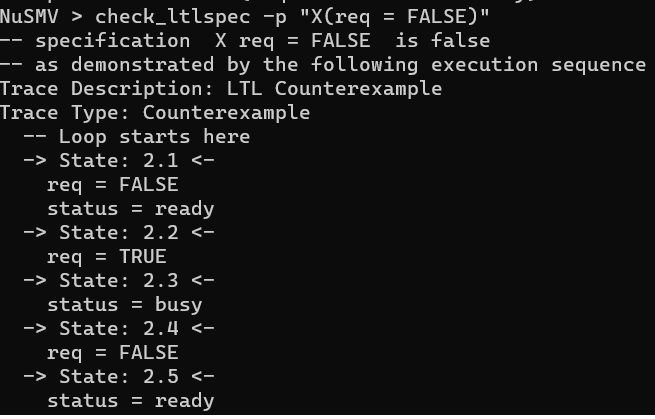
Description automatically generated

**PART-D**

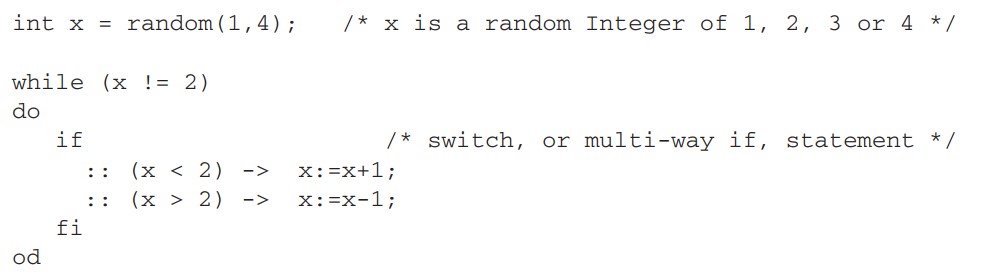
1. **G(request -> F status=busy) :**

****

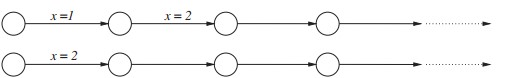
1. **X(request = FALSE):**

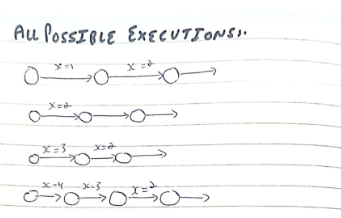
****

**TASK # 03:** Consider a program (given in a C-like language)

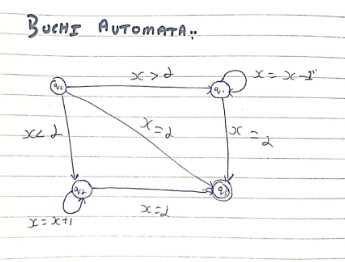


1. Write all possible (temporal) executions of this program (2 executions are mentioned here)

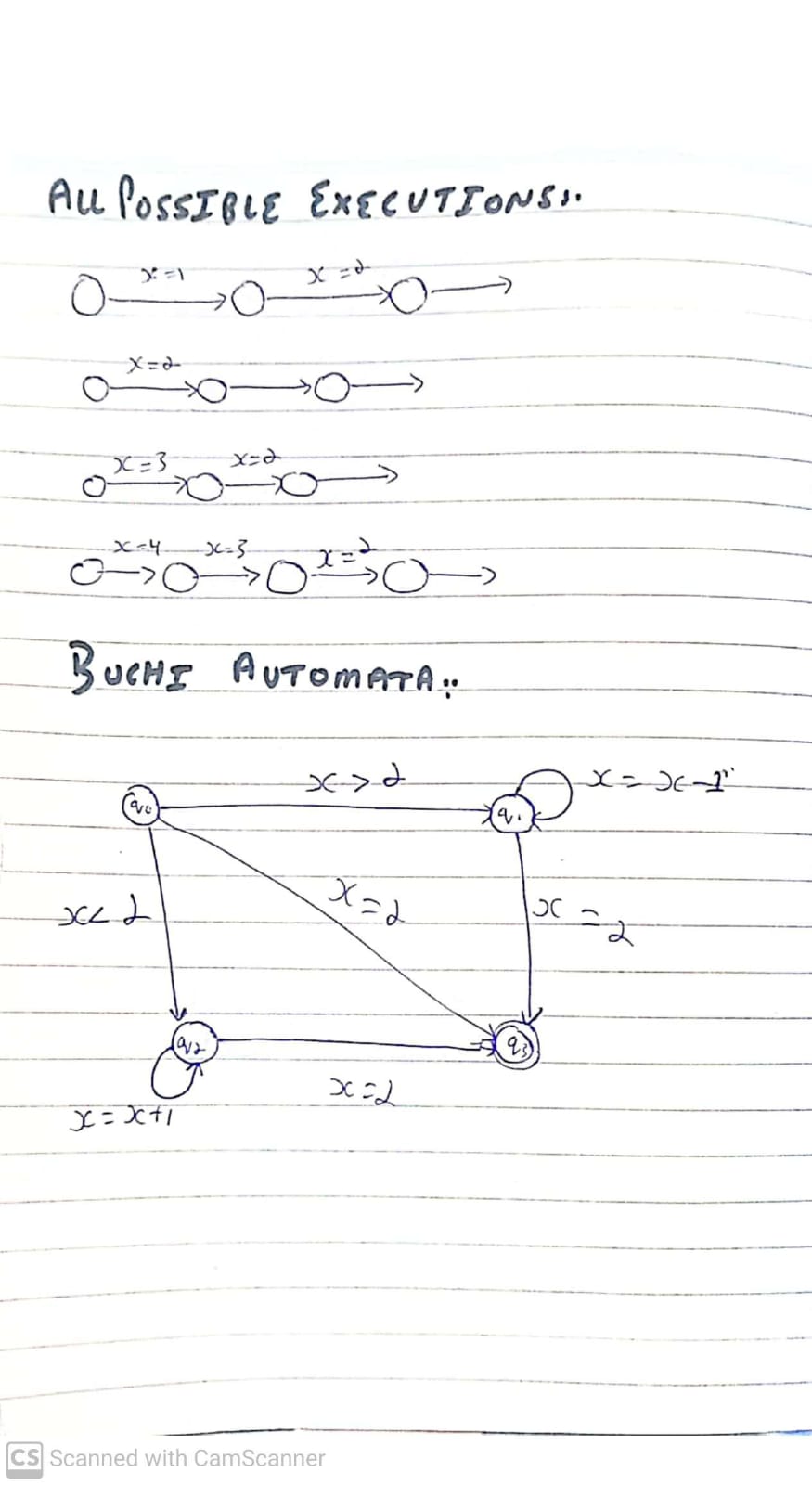




1. Construct a Büchi Automaton representing all executions of the above program.



Refrence:



xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx