UNIVERSITY of ALBERTA
Department of Electrical and Computer Engineering

ECE 321 Software Requirements Engineering

FINAL EXAMINATION

9:00 AM, December 14th, 2015
Time: 120 min
100 pts TOTAL

NAME (PRINT NEATLY):_____

DO NOT OPEN EXAM UNTIL INSTRUCTED TO DO SO.
READ QUESTIONS CAREFULLY.
ATTEMPT ALL QUESTIONS.
ANSWER QUESTIONS ON THE EXAM PAPER.
ANSWER CROSSED OUT WILL BE IGNORED.
GIVE ONLY ONE ANSWER – ALL SUBSEQUENT ANSWERS WILL BE IGNORED

NOTES:

- **Formula Sheets, Calculators, or any other materials are not allowed**
- **Pay particular attention to sentences and words in bold**
- **Good luck!**

part	max points	**received points**	topic
I	40		Multiple choice and matching table questions
II	30		Theory
III	30		Hands-on Exercises

Part I. Multiple choice and matching table questions (total of 40 pts)

- 1 matching table (8 pts), 20 multiple choice questions (1 pts/each), and 12 T/F questions (1 pt/each)
- each multiple choice and T/F question has only ONE correct answer; circle the correct answer
- the solution to the matching table consists of correct matching between the phrase number and the description letter

1. (*8 pts*) Match the phrase with its description:

Phrase		Description	
1	usability	**A**	an approach to prune, group, and prioritize input collected at a brainstorming workshop session
2	reusability	**B**	concerns ease of use (user–friendliness) of a software product
3	storyboarding	**C**	integral part of a vision document
4	idea reduction	**D**	concerns ability to convert a software component for use in other applications
5	exception	**E**	passive, active, or interactive technique to implement a session during a workshop
6	invariant	**F**	sketch or drawing of a user interface
7	horizontal prototype	**G**	one of elements utilized by the design by contract methodology
8	business objective	**H**	one of elements of a fully dressed use case

1	2	3	4	5	6	7	8

1. *(1 pt)* Risk analysis is one of the key concepts introduced in the following software life cycle model:
   a. Waterfall
   b. Extreme Programming
   c. Spiral

2. *(1 pt)* Which of the following is an **internal** software quality?
   a. testability
   b. availability
   c. usability

3. *(1 pt)* The customer's statement that says "The software should implement the process of ordering a chemical" would **most likely** lead to
   a. defining a use case
   b. defining a non-functional requirement
   c. defining a business requirement

4. *(1 pt)* **Feasible** requirement statement is
   a. possible to verify via testing
   b. possible to implement
   c. linked to the corresponding design, source code and test cases

5. (*1 pt*) Software requirements are developed using the following sequence of steps (note the **order** of the steps):
   a. use cases; elicitation; validation and verification
   b. elicitation; specification; and validation and verification
   c. specification; elicitation; and validation and verification

6. *(1 pt)* The complete list of **direct** stakeholders includes
   a. Requirement analysts
   b. Customers and users
   c. Sales, marketing, and legal staff

7. *(1 pt)* The **development** of the requirements, in contrast to the **management** of requirements, includes:
   a. identifying the stakeholders
   b. tracking the status and change of the requirements thorough the project lifetime
   c. tracking individual requirements to their design, code, and tests

8. *(1 pt)* The **last** step in the 11-step interview aiming at the elicitation of user needs is the
   a. brainstorming session
   b. recap for understanding
   c. interviewer's summary

9. *(1 pt)* Which of the following is **not** included in the use case?
   a. priority
   b. a normal course (also called primary scenario)
   c. axioms

10. *(1 pt)* Which of the following prototype types would be **the least** suitable to serve as the evolutionary prototype?
   a. vertical prototype
   b. throwaway prototype
   c. horizontal prototype

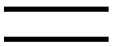
11. *(1 pt)* The Vision document should be developed
   a. **before** the Software Requirements Specification document
   b. **before** identifying the stakeholders
   c. **after** completing the Software Requirements Specification document

12. *(1 pt)* Which of the following **is** a part of a Software Requirements Specification document?
   a. test cases
   b. budget
   c. nonfunctional requirements

13. *(1 pt)* The IEEE 830 standard for the Software Requirements Specifications includes the following section
   a. vision statement
   b. budget
   c. product perspective

14. *(1 pt)* The **descriptive** specifications are concerned with
   a. properties of a software system, and are usually given using axioms (statements of properties) or algebras (sets of operations and values)
   b. behavior of a software system (sequences of states), and are usually given using an execution model
   c. relationships in a software system, and are usually given using multi/hyper-graphs

15. *(1 pt)* Which of the following models suffers from the **state space explosion**
   a. RAISE
   b. Finite State Machines
   c. Petri Nets

16. *(1 pt)* In the Data Flow Diagrams, the **data stores** are represented by
   a.  b.  c. 

17. *(1 pt)* A Finite State Machine is defined by
   a. Finite set of states, finite set of places, and finite set of inputs
   b. Finite set of places, finite set of transitions, and transition function
   c. Finite set of states, finite set of inputs, and transition function

18. *(1 pt)* A Petri Net is said to be **1-bounded** if
   a. None of its places has ever more than 1 token
   b. Every transition in this network has 1 input and 1 output place
   c. The network has never more than 1 token across all of its places

19. *(1 pt)* The keywords used in the **algebraic specifications** include
   a. type, value, and variable
   b. precondition, postcondition, and invariant
   c. sort, signature, and generator

20. *(1 pt)* In the concept ‘design by contact’, a contract is ‘set’ between
   a. two pieces of software
   b. two programmers
   c. employee and employer

1. *(1 pt)* (True/False)
   Gold plating refers to the inclusion of additional functionality by a developer who thinks that the user/customer will need and like this functionality.

2. *(1 pt)* (True/False)
   Analysis of **root causes** is helpful for the writing of the Vision document.

3. *(1 pt)* (True/False)
   Petri Nets and Finite State Machines are examples of descriptive (also called declarative) specification models, while Algebraic specification is a structural model.

4. *(1 pt)* (True/False)
   Vertical prototyping can be used to elicit requirements.

5. *(1 pt)* (True/False)
   Use cases describe **how** users want a system to be built.

6. *(1 pt)* (True/False)
   The Vision Document provides **detailed** description of the business objectives.

7. *(1 pt)* (True/False)
   IEEE Standard 830 includes sections that explicitly address several software qualities including reliability, availability, and security.

8. *(1 pt)* (True/False)
   The IEEE 830 Standard explicitly asks to define system, user, hardware, software, and communication interfaces.

9. *(1 pt)* (True/False)
   In Finite State Machines, the double-circled nodes  denote states with at least 2 incoming transitions (i.e., at least 2 states will transition into the double circled states).

10. *(1 pt)* (True/False)
    It is **possible** to define a valid Petri Net that cannot be represented by a Finite State Machine.

11. *(1 pt)* (True/False)
    Petri Nets can express systems with an **infinite** number of states.

12. *(1 pt)* (True/False)
    In the conpcet ‘design by contract’ invariants are used to check if variables have always valid values.

Part II. Theory (total of 30 pts)

- 4 theoretical questions (three at 8 pts and one at 6 pts)
- please provide an answer within the allocated space and write clearly and neatly!

1. *(8 pts)* Briefly (using one sentence) **define** robustness and reliability software qualities. Give **two robustness requirements** for the Traffic Lights System that was discussed in the labs.

definition of robustness

definition of reliability:

first example of robustness-related requirement for the Traffic Lights System:

second example of robustness-related requirement for the Traffic Lights System:

2. *(6 pts)* One of the initial tasks in the development of a software requirements specification document is the "analysis of the problem". This task is implemented using 5 steps, with the first step being "gaining an agreement on the problem definition". You must **name at least 3 out of 4 of the remaining steps**; do not (re)list the above step. The steps do not have to be in order and you can list 4 steps and the best 3 among them will be marked.

1. ……

2. ……

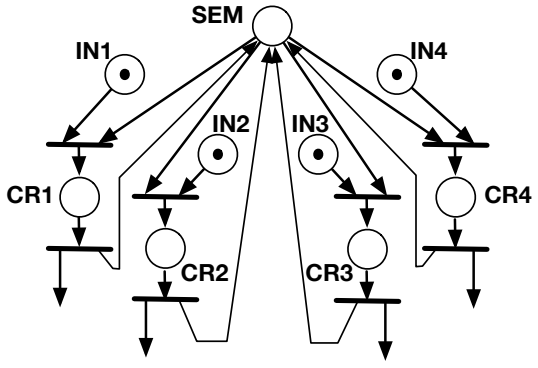
3. ……

4. ……

3. *(8 pts)* Use cases are documented using several fields. You must correctly identify which of the following fields **are** and **are not** included in a use case template.

	write "yes" if the field is included in a use case; otherwise write "no"
unique identifier	
list of preconditions	
list of postconditions	
software interfaces	
normal course (primary scenario)	
safety requirements	
priority	
actors	
security requirements	
frequency of use	
business rules	
assumptions	
exceptions	
data definitions	
horizontal prototypes	
glossary	

4. *(8pts)* The Petri Net shown below could be a model of a semaphore that controls access to four critical parts $\mathbf{CR_1}$, $\mathbf{CR_2}$, $\mathbf{CR_3}$ and $\mathbf{CR_4}$. How many tokens (could be 1, 2, 3, 4, 5 and so on) are required in the place **SEM** to model such a situation? Please describe behavior of a system modeled by this Petri Net when there are 3 tokens in the place **SEM**.

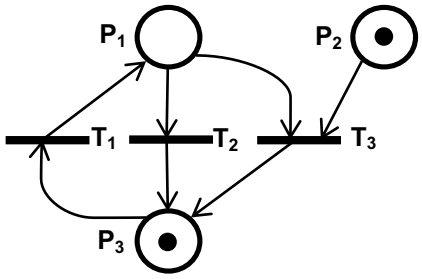


Part III. Hands-on Exercises (total of 30 pts)

- 3 hands-on exercises (10 pts each)
- please provide an answer within the allocated space and write clearly and neatly!

1. *(10 pts)* Provide a **Finite State Machine** (FSM) model that is equivalent to the below Petri Net. Your solution must include
- **explanation/definition of all states** from your FSM using the terminology introduced in the Petri Net model
- the **graph** of the FSM (you must name/enumerate states and inputs in the graph)
- the **state transition function table** in which inputs are in columns and states in rows

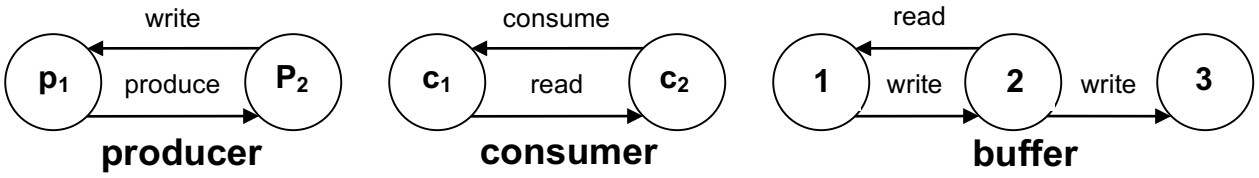
The model must use names of the inputs that are given in the Petri Net.



2. *(10pts)* Neatly draw a **Finite State Machine** that models a producer-consumer system by combining behavior of the given below Finite State Machines for the producer, consumer, and buffer objects. You must follow these assumptions:

- The producer produces items, which are written into a buffer; consumer reads items from the buffer and consumes them.
- Producer's write is the same as buffer's write; consumer's read is the same as buffer's read.
- The buffer has limit of 3 items and always has at least 1 item. The system **stops** after 3 items are stored in the buffer; you must **show** this on the model.

You must describe the system state associated with each node in your model, i.e., you should use the "p_1","p_2","c_1","c_2","1", "2", and "3" to do that. Remember to name the inputs in your model. Use back of an adjacent page if you like to first draft your solution.



3. *(10 pts)* **Fill in** the missing parts (**definition of operations**, **names of generators**, elements associated with the **all** statement, and **axiom** for the *NumberOfCars* operation) for the below algebraic specification. Be **precise** in terms of syntax and use only symbols introduced in the algebra. Note that the definition of *New* is already provided and that *NumberOfCars* should compute number of Cars for the non-empty Queue.

algebra QueueOfCars
imports Boolean, Integer;
introduces
sorts Queue, Car;
operations
New: $\rightarrow$ *Queue;*

……

……

……

……

……

constrains New, CarArrives, CarDeparts, IsEmpty, NumberOfCars, Longer so that Queue

generated by [……]

for all [……]

CarDeparts(New) = New;
CarDeparts(CarArrives(c1, q1)) =
if (IsEmpty(q1) = = true) then New;
else CarArrives(c1, CarDeparts(q1));

IsEmpty(New) = true;
IsEmpty(CarArrives(c1, q1)) = false;

NumberOfCars(New) = Error;
NumberOfCars(CarArrives(c1, q1)) =

……

……

Longer(New, i1) = false;
Longer(CarArrives(c1, q1), i1) =
if (NumberOfCars(q1) = = i1)
then true;
else Longer(q1,i1);

end QueueOfCars;