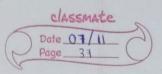
## TUTORIAL - 788



| Q.1.7  | Write a Progrem Specification to compute greatest common divisor.  |
|--------|--|
|        | Pre-condition (is to and is to y   |
|        | Procedure P  |
|        | Past-condition 0= Z and such that  |
|        | exist $z$ , $k_1$ , $k_2$ ( $i_1 = z \times k_1$ and $i_2 = z \times k_2$ )  |
|        | not exist y (i1 = $y \times k_3$ and i2 = $y \times k_4$ and   |
|        | Cexist K3, K4)   |
| 000    | lating a language of the same  |
|        | Worke a Program Specification to produce reverse of input sequence.  |
| lloa T | Precondition of no fit of the part of the standard from the first and  |
| - 2    | Procedure voto Postantia de la company topolo de la |
|        | Post-condition for all is $C \in \mathbb{N}$ , $O \in \mathbb{N}$ , $O \in \mathbb{N}$   |
|        |  |
| 9.3.)  | Give a logic specification of for a prot program that reads a sequence of N+1 values and checks whether the first  |
| A3.)   | value also appears in next input n values.   |
| 113.)  | check Element ( n, io, li, in's)   |
|        | o impuce ( exist j ( 1 \le j \le n) and io = ij )  |
|        | 3  |

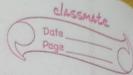
Give a logic specification for a program that first red two words (i.e. two sequences of alphabetic enarative by a blank and terminated by '#'). The second word may be null, the Axt must not. Then, the program reads a sequence of other words, seperated by blanks and terminand by "#", and rewrited the sequencel, Substituting that all occurrences of first word by second, PIG

replace Word ( n, woods, nz, sentence)

( 0-sentence = '\pm')

n3 × 0

4



|       | Systems   |
|-------|---|
| Q 5.> | Mention advantages and disadvantages of using FSM specifications for      |
| ASY   |   |
|       | Advantages of FSM   |
| 11/1  | AS-DIF a chosen ) In sist on i do of                                      |
|       | 1 FSM simplicity make it easy for inexperienced developers to             |
|       | implement with little to no extra knowledge (low entry lend)              |
|       | 2 Predictability (in deterministic FSM)                                   |
|       | 3) Quick to design, Quick to implement & Quick in Execution               |
|       | @ Relatively Flexible. (Number of ways to implement FSM)                  |
|       | 6 Easy to transfer meaningful abstract representation to a                |
|       | coded implementation.   |
|       | @ Low processor overhead of easy determination of reachabitity of         |
|       | Diaduation of tem   |
|       | Disadventages of F3M  |
|       | 1) The predictable nature of deterministic F3M can be difficult to manage |
|       | Some domains like computer gomes  |
|       | ② Finite memory  > Expressive Power is limited                            |
|       | 3 State Explasion   |
|       | y circen a number of FSMs with ki, key kn states their                    |
|       | composition is a FSM with k1 x kx xkz kn. (Exponential)                   |
|       | a) FSMs are essentially a synchronous model ( at any time)                |
|       | a global state of the system, must be defined and a single                |
|       | transition must occur.  |
|       |   |
|       |   |
| -     |   |

067 Write Algebraic Specification for Stack operations. algebra Stack of Item imports Booken; introduces sort stack, Item; Handle and March operations Create: → Stack; Stack create() → Stack Is Empty: Stack -> Boolean; push C Stock, Integer) -> Stock Push: Stack x Iron -> Stack; pop ( Stock) -> Stack Ap: Stack -> Stack; peck (Stock) -> Integer Top: Stack > Item: Is Emply (Stock ) - 1 Robbon constraints Creat, Is Empty, Push, Pop, Top so that Stack generated by [create, Push] for all [ S: Stack, i: item] Pop(create()) = (Exception (empty stock) Teempty (Create) = true; pop ( push (s, I)) = S Is Empty ( Push (s, i)) = false; peck (create()) = exception ( empty Pop ( Create) = error; Top (Create) = error; peck c push (S, I) = I Pop ( Push (S, 1)) = S; is Empty ( creaters) = true Top ( Push (s, i) ) = i; IS Empty ( Push (S, E)) = Pale end Stack of Iten; I am an analysis and

Compare or Differentiate PetriNets with FSMg Y.FO We know that both Petrinets and FSM are models that represent ATY the discrete interactions in system. Both are not possible practically but define the behavior of system. Here the difference b/w Petrinets CPN) and Finite State machines CBMI 1) FSM is a model that represent how single activity change its behavior over time wherelse PN approxents multiactivity co-ordinates. 1 In FSM, designer knows the starting Point and how the process is going, but PW is asymptonous process in which designer doce not know about storing point and how process is working Concurrent FSM occurs only on one system based on time, but PN contain co-ordinate system that included two or three processes, one depend on the provious one. So deadlock conbe placed in PN. TSM can be based on inputs and outputs and of two types - Mealey FSM and Moore FSM but PN is based on places ( like communication votes) transition (like bronsformation of objects) and lakens (like physical objects 1 Petrinute are suited for timed processes wherelse FSMs are not suited for it.