|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **GOVT. MODEL ENGINEERING COLLEGE, THRIKKAKARA (Managed by IHRD, A Govt. of Kerala Undertaking)**  **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING** | | | | | | | | | | | |
| **B.TECH. DEGREE FIFTH SEMESTER EXAMINATION**  **COMPUTER SCIENCE AND ENGINEERING**  **SECOND INTERNAL EXAMINATION – NOVEMBER 2019** | | | | | | | | | **Academic Year:**  **2019-20** | | | | |
| Slot : **A** | | Course Code: **CS 301** | | | Course Title: **THEORY OF COMPUTATION.** | | | | | | | | |
| Duration: 1 Hr. | | | | Max. Marks: 20 | | Faculty Handling the Course: **VINEETHA K V** | | | | | | | |
| ***Course Outcomes:*** *At the end of the course the students will be able to* | | | | | | | | | | | | | |
| *CS301.1* | | | *Classify formal languages into regular, context- free, context sensitive and unrestricted languages.* | | | | | | | | | | |
| *CS301.3* | | | *Design push-down automata and context-free grammar representations for context-free languages.* | | | | | | | | | | |
| **(****Answer All Questions)** | | | | | | | | *Cognitive level* | | | ***CO*** | | **Marks** |
| 1 | Using pumping lemma prove that the set L = { 0n | n is a perfect square } is not regular. | | | | | | *Applying* | | | *CS301.1* | | 3 | |
| Ans | Pumping Lemma :  Let L be a RL. Then there exists a constant n s.t.,if w ∈ L s.t. |w| ≥ n, then we can write w=xyz such that: 1.|xy| ≤ n 2.|y| ≥ 1 3.For all i≥0: xyiz ∈ L  Proof:  Let w = 0n where n = k2 |w| = |xyz| = |0n  | =n = k2 let us take |y| = k for i=2 :length of |xy2z| must be the next perfect square , that is (k + 1)2 .  But |xy2z| = |xyz|+|y| = k2 + k < (k + 1)2 .  for i = 2, |w| is not a perfect sqaure. So L is not a Regular Language. | | | | | |  | | |  | | 1+2 | |
| 2 | Check whether the grammar is ambiguous or not.  S → aB | bA  A → a | aS | bAA  B → b | bS | aBB | | | | | | *Applying* | | | *CS301.1* | | 3 | |
| Ans | A CFG is said to be ambiguous if there exists a string which has more than one left-most/right-most derivation.  Consider input string aabbab  S => aB => aaBB => aa**b**B **=>** aab**bS** => aabbaB => aabbab  S => aB => aaBB => aa**bS**B => aab**bA**B => aabbaB => aabbab  S S  / \ / \  **a** B  **a** B  / | \ / | \  **a** B B **a** B B  | / \ / \ |  **b b**  S  **b** S **b**  / \ / \  **a** B **b**  A  | |  **b a** | | | | | |  | | |  | | 1+2 | |
| 3 | Design a PDA to accept the language L = { w | na(w) = nb(w) } ( Note : na(w) represents number of a's in w) | | | | | | *Applying* | | | *CS301.3* | | 3 | |
| Ans | PDA M = ( { q0 }, {a,b}, {X,Y,Z0}, δ, q0 ,Z0 , Φ)  δ(q0,a,Z0) = { (q0, XZ0) }  δ(q0,b,Z0) = { (q0, YZ0) }  δ(q0,a,X) = { (q0, XX) }  δ(q0,a,Y) = { (q0,ε ) }  δ(q0,b,X) = { (q0, ε) }  δ(q0,b,Y) = { (q0, YY) }  δ(q0,ε,Z0) = { (q0, ε) }    Input string aababbba: Accepting  ( q0 , aababbba , Z0 ) ├ ( q0 , ababbba , XZ0 ) ├ ( q0 , babbba , XXZ0 ) ├ ( q0 , abbba , XZ0 ) ├ ( q0 , bbba , XXZ0 ) ├ ( q0 , bba , XZ0 ) ├ ( q0 , ba , Z0 ) ├ ( q0 , a , YZ0 ) ├ ( q0 , ε , Z0 ) ├ ( q0 , ε , ε )  **(Not Necessary)**  Input string aababbb: Rejecting  ( q0 , aababbb , Z0 ) ├ ( q0 , ababbb , XZ0 ) ├ ( q0 , babbb , XXZ0 ) ├ ( q0 , abbb , XZ0 ) ├ ( q0 , bbb , XXZ0 ) ├ ( q0 , bb , XZ0 ) ├ ( q0 , b , Z0 ) ├ ( q0 , ε , YZ0 )  No transition for ( q0 , ε , YZ0 ) . so string is not accepted by PDA. | | | | | |  | | |  | | 2+1 | |
| 4 | Convert the following CFG to CNF  G = ( {S, A}, {a, b}, P, S)  where P = { S → AbA  A → Aa | ε } | | | | | | *Applying* | | | *CS301.3* | | 3 | |
| Ans | First eliminate ε production.  Productions will be S → AbA | Ab| bA | b and A → Aa | a  CNF  S → ACbA  S → AX1 X1 → CbA  S → ACb  S → Cb A  Cb  → b  A → ACa |a  Ca→ a] | | | | | |  | | |  | | 3 | |
| 5 | Simplify the following grammar  S → Aa | B  B → a | bC  C → a | ε | | | | | | *Applying* | | | *CS301.3* | | 4 | |
| Ans | Simplification CFG  1. Eliminate Null productions  2. Eliminate Unit productions  3. Eliminate Useless symbols  1. Eliminate Null Productions  C is the only Nullable variable in the grammar.  After eliminating null production , Grammar will be  S → Aa | B  B → a | bC | b  C → a  2. Eliminate Unit productions  S → B is the only unit production. After removing unit production grammar becomes  S → Aa | a | bC | b  B → a | bC | b  C → a  3. Eliminate Useless symbols  i) Eliminate variables which are not deriving/genarating any terminal symbols.  A is not generating any terminal strings.  S → a | bC | b  B → a | bC | b  C → a  ii) Eliminate variables which are not reachable.  B is not reachable.  S → a | bC | b  C → a  Simplied Grammar is  S → a | bC | b  C → a  . | | | | | |  | | |  | | 1+1+1+1 | |
| 6 | Convert the following CFG to PDA with empty stack  S → aA  A → aAB | aA |a  B → b  Write Instantaneous Descriptor for the acceptance of the string 'aaab'. | | | | | | *Applying* | | | *CS301.3* | | 4 | |
| Ans | If G= (V,T,P,S) we can construct PDA such that M = ({q}, T, V U T, δ, q, S, Ø)  where δ is  i) For all A Є V , add the following transition(s) in the PDA: δ(q ,ε ,A) = { (q, α) | “A → α ” Є P}  ii) For all a Є T, add the following transition(s) in the PDA: δ(q,a,a)= { (q, ε ) }  S → aA == > δ(q ,ε ,S) = { (q, aA) }  A → aAB ==> δ(q ,ε ,A) = { (q, aAB) }  A → aA ==> δ(q ,ε ,A) = { (q, aA) }  A → a ==> δ(q ,ε ,A) = { (q, a) }  B → b ==> δ(q ,ε ,B) = { (q, b) }  For input a δ(q,a,a)= { (q, ε ) }  For input b δ(q,b,b)= { (q, ε ) }  Input string :aaab  ID : (q , aaab, S) ├ (q , aaab, aA) ├ (q , aab, A) ├ (q , aab, aAB) ├ (q , ab, AB) ├ (q , ab, aB) ├ (q , b, B) ├ (q , b, b) ├ ( q , ε , ε ) | | | | | |  | | |  | | 2+2 | |