Left Recursion & Left Factoring Assignment: D Program > programid (identifiers_list); declarantings Subprogram Compound-statement. bothings to Remove 2) Identifier-list > id | identifier-list, id Removing Left Recursion identifies-19st > id identifies_19st identifier-195t -> , id Paentifier-185t / E 3) declarations -> declarations var identifies-list: type; Removing Left facursion declarations > vor identifiers-list: type; declarations declarations -> declarations' 4) type > standard-type | arosay [sum.... sum] of standard-type 5) Standard-type > integrer / real 6) suprogram-declarations > suprogram declaration ; E Removing Left Recursion Subprogram - declarations - subprograms - declarations | E Subprogram-declarations > subprogram-declarantion ? subprogram-declarations' \ E 7) supragram-declaration - s supragram-head declarations compound-statement 8) subprogram-head -> function id orgaments: standard type; 8) adjuments > (poormeter 19t) / E 19) pasameter-list - identiffer-list; type | parameter-list; identificalist: HPR Removing Left Rocusifus prometer-19st -> identifier-19st: prometer_19st Parameter-19st's; identifier-19st: type parameter 19st |E

10) subpragram had a function id arguments: standard-type; procedure id organients; Removing Loft Factoring watthing to Remove. Subprogram had id arguments subprogram had go ; stordood- type ; 11) compound-statement -> begin optional statement end 12) optional_statement > statement_list | E 13) Statement-list > Statement | Statement-list; Statement Startement-11st -> Statement startement-11st ! Removing Left Recubsics stalement-19st' >; statement statement-11st' | E 14) Statement -> variable assignop expression | procedure-statement | compound-statement | if expression then statement else statement | while expression do statement. 15) varoiable > 81 / 81 (exprosssion) Removing left Factoring versiable > id variable versiable -> [expression] / E 16) expression_list > empression | expoession_list, expression empression-list - s empression empression-list Removing Left Recursion expossion-list >, expossion expossion-list | & Removing Left Factoring exposures exposures exposures exposures exposures empoession > relap Apaple-empoession | E

18) sprople-expossible > ters | signateons | simple-expossion addop toom Removing Left Recursion

semple-exposures -> addop toom semple-exposures 'E

simple-exposures -> addop toom semple-exposures 'E 19) toom > factor I toom mulop factor Removerey Left Recooses team > factor team team' > mulop factor team' (& 20) factor-sid lid (expression-list) | num | (expression) | Removely left fectoring fortes -> 30 fortes, I want (extraction) and togo factors' => (exposuren-19st) 1 & segn >+1-Nothing to Remove 22) Procedure-Statement -) Pol (enpression-list) Procedure-Statement -> id Procedure-Statement Romourg Left Foctooring Proceeduse-Statement'> (exprosser-list) / E