Fourish Early this applicant

CS 4323 PROGRAMMING ASSIGNMENT #2 (PARSER)

Due: April 26, 2018

from HW #1, 5

Refer to the lexical definition of TrumpScript++ Language (A1), the syntax rules for TrumpScript++ language (A2), the LL(1) parse table for TrumpScript++ (A3), and the general structure of the LL(1) parser (A4). Add to your compiler a procedure/class PARSER() which, when given an arbitrary input program, generates its parse output, using the syntax rules for TrumpScript++ and the LL(1) parse table.

In order to write PARSER(), the first thing you have to do is to transform the syntax rules for TrumpScript++ (given in A2) into a PDA, which is nondeterministic in general. The transformation algorithm was given in class. Now, with the lookahead of one token allowed (note that a token is viewed as a single symbol here) and the LL(1) parse table (given in A3) added to the finite control, the PDA is a deterministic LL(1) parser.

PARSER() should be called *once* from the main body of your program. Once it is called from the main body, the whole parsing must be done in PARSER().

SCANNER() needs some modification now. First, it will not be called from the main body (see A4); it will be called from PARSER() when a lookahead token is needed. Second, your previous SCANNER() printed a token as it identified one; here, it must pass it to PARSER(). SCANNER() will pass it in the form of a pair (token, token type), where token means the actual token identified (i.e., lexeme) and token type means the classification of tokens into those represented in the syntax rules. Finally, SCANNER() need not print out the scanning information now, since you know that it does the correct job (if you correctly constructed SCANNER() in the first programming assignment).

It is convenient to use *integer codes* to classify token types. We shall use integers 1..3 for [id], [const] and [string], integers 4..26 for the 23 keywords, and integers 27..33 for the 7 special symbols.

It is also convenient to define the stack of the parser using an array of integers; an array of size 100 should be enough for our purpose. Note that the stack stores nonterminals and terminals. For terminal tokens, use the integer codes 1..33 passed from SCANNER(). Use integers 34..53 for the 20 nonterminals in the grammar. Use 0 for the stack bottom marker.

More details will be discussed in class, with examples. For the required output, the following informations must be printed in a well-documented form:

Print out the source program to be parsed (given on the next page), exactly as you stored in your input file. This must be done before parsing begins.

• Print out the parse output, i.e., the sequence of actions taken by the parser together with supporting informations (stack-top symbols and lookhead tokens in their original names and integer codes). An example of the parse output will be given in class; it is important to follow the output format given in this example.

 Print out the SYMTAB content. This must be done in the main body of your compiler after parsing has been completed.

Your compiler after

Your compiler after

Your sor did not heart what head exactly it i

scanner will "solver if the lookahead is consumed exact)

called the frequency of the lookahead is consumed exact)

source. edithed dup has. chuned from heline war a pux Source program: Make programming great again # main body begins Make x number; make y1 z2zz w number; make a b Boolean; X is 1000001; y1 is 2000000; z is 123456789; A is fact; b is lie; As long as, fact or lie; If, more x plus (y) times 2000000 z?; tell a b; say "stop"; ! else make c Boolean; !(;)
C is not not not fact and less x z ?;
Tell a b c x y z;
Say "done"; # say done
!(;)
America is great Tell x y1 z2zz; say "continue"; Sowher Submit a hard copy of your program and output on the due date in class. An incomplete work claiming partial credit (of no more than half the full credit) should include on the first page of the submitted work a description of what troubles you had. Failure to include this information may result in zero credit on the Stack + luskahead of bottom marked 10 / Latile. Expands work. Penalty for late submission: 10% per calendar day. Assignment handed out on April 3, 2018. Sample output LL(1) parse output Stack top symbo ahead 20(0) Ande Souther Trump (untial) Scannor Drodramming match a clair Watch Make. rule Yule (12) make ιO Nake ny whel 200 51285



A1. Lexical definition of TrumpScript++ Language ike, programming, great, again, america, is, else, number, boolean, if, as, long, Keywords: ma tell, say, fact, lie, not, and, or, less, more, plus, times. • Identifiers: any letter followed optionally by digits and/or letters. • Constants: any sequence of digits whose corresponding value is greater than 1,000,000. A2. Syntax Rules for TrumpScript++ Language May Cfirst > stmts > clast | push | cfirst > Make programming great again [3] < last > America is great stmts> -> <stmt> ; <more-stmts> [5-6] <more-stmts -> <stmt>; <more-stmts> | \epsilon [7-11] <stmt> -> <decl> | <asmt> | <cond> | <loop> | <output> [12] <decl> -> make <ids> <type> [13-14] <type> -> number | boolean [15] <asmt> -> [id] is <expr> [16] <cond> -> if, <bool> ; : <stmts> ! else : <stmts> ! [17] <loop> -> as long as , <bool> ; : <stmts>! [18-19] <output> -> tell <ids> | say [string] [20] <ids> -> [id] <more-ids> [21-22] <more-ids> -> [id] <more-ids> | \epsilon [23-24] <expr> -> <bool> | <arith> [25-28] <bool> -> fact <bool-tail> | lie <bool-tail> | not <bool> | <test> <arith> ? [29-31] <bool-tail> -> and <bool> | or <bool> | \epsilon [32-34] <test> -> less | is | more [35-37] <arith> -> [id] <arith-tail> | [const] <arith-tail> | (<arith>) <arith-tail> [38-40] <arith-tail> -> plus <arith> | times <arith> | \epsilon

< Trampo

A3. LL(1) Parse Table for TrumpScript++ Language

(more-stmits) 5 FIRST, ((stmt)) America! FIRST, ((stmt)) 6 E America! 7 (make)	<u> </u>				
(first) 2 Make FIRST, ((stmt)) / Make (last) 3 America & / America (stmts) 4 FIRST, ((stmt)) America! / FIRST, ((stmt)) (more-stmts) 5 FIRST, ((stmt)) America! / America! 7 (make) America! / America! / America! 8 Cid1 (Id1) (Id1) (stmt) 9 if if if 10 as as 11 tell say tell say (decl) 12 (make); make (type) 14 boolean in boolean (asmt) 15 [id1]; (if (loop) 17 as if (loop) 17 as if (loop) 17 as if (loop) 18 tell (more-ids) 20 [id1] mumber boolean (more-ids) 21 [id1] mumber boolean (asmt) 22 & E 12 FIRST, ((bool)) FIRST, ((bool))	Productions		FIRST,	FOLLOW,	LOOK AHEAD,
(Starts) 3 America & America (Starts) 4 FIRST, ((Starts) America ! FIRST, ((Starts)) 6 & America ! FIRST, ((Starts)) 6 & America ! FIRST, ((Starts)) 6 & America ! [Id] (Starts) 9 if	(Trump)	1	Make	3	Maker
(stmts) 4 First, ((stmt)) America! First, ((stmt)) (more-stmts) 5 First, ((stmt)) America! First, ((stmt)) (make) 8 Cid1 Cid1 (stmt) 9 if if if 10 as as 11 tell say tell say (decl) 12 (make) ; make (type) 14 boolean if boolean (asmt) 15 [id] ; [id] (Cond) 16 if if (Loop) 17 as if (asmt) 18 tell if (autput) 19 say (id) 10 mumber boolean (more-ids) 21 Cid] mumber boolean (ever) 23 First, ((bool)) if if (cool) 12 mumber boolean (asmt) 22 E	(first)	2	Make	FIRST, ((stmt))	Make
(more-stricts) 5 FIRST, ((strict)) America! FIRST, ((strict)) 6 E America! 7 (make) America! America! 7 (make) Make 8 Cid1 Cid1 (strict) 9 if; if 10 as as 11 tell say tell say (decl) 12 (make); make (type) 14 boolean; boolean (asmt) 15 [id]; Cid1 (Cond) 16 if; if (Loop) 17 as; as (output) 18 tell (output) 19 say (ids) 20 [id] mumber boolean (more-ids) 21 Cid1 (more-ids) 22 E 3 FIRST, ((bool)) FIRST, ((bool))	(last)	3	America	٤	/America
(stmt) G E America! Make Make Make	(stmts)	4	FIRST, ((stmt))	America!	FIRST, ((stint >)
Cmake make make	Convact to	5	FIRST, ((stmt)	1	FIRST, ((stint))
(stmt) 9 if ; if 10 as as 11 tell say tell say (decl) 12 (make) ; make (type) 14 boolean ; boolean (asmt) 15 [id] ; [id] (Cond) 16 if ; if (Loop) 17 as ; as (output) 18 tell ; tell (more-ids) 20 [id] mumber boolean (asmore-ids) 22 E ; FIRST, ((bool))	Knight G-7 (MC)	16	3	Fimerica:	America!
(stmt) 9 if ; if 10 as as 11 tell say tell say (decl) 12 (make) ; make (type) 14 boolean ; boolean (asmt) 15 [id] ; [id] (Cond) 16 if ; if (Loop) 17 as ; as (output) 18 tell ; tell (output) 19 say ; say (ids) 20 [id] mumber boolean [id] (more-ids) 21 [id] mumber boolean [id] (more-ids) 22 E ; mumber boolean (ever) 23 FIRST, ((bool)) ; FIRST, ((bool))		7	make		make
10 as as tell Say tell Say (decl) 12 (make) ; make make mumber mumber type 14 boolean boolean boolean Casmt 15 [id] ; [id] (Cond) 16 if ; if (Loop) 17 as ; as (asput) 18 tell ; tell ; asy (ids) 20 [id] mumber boolean cid] (more-ids) 21 Cid] mumber boolean cid] (more-ids) 22 E ; mumber boolean cid] (ever) 23 FiRST. ((bool)) FiRST. ((bool))		8	Cid1		Lidj
(decl) 12 (make); make (decl) 12 (make); make (type) 13 number (type) 14 boolean; boolean (asmt) 15 [id]; [id] (Cond) 16 if; if (loop) 17 as; as (output) 18 tell; tell (output) 19 say; say (ids) 20 [id] number boolean (more-ids) 21 [id] number boolean (every) 23 FIRST, ((bool)) (every) 18 tell field (fill) number boolean (every) 23 FIRST, ((bool))	(stmt)	9	îf	· ;	;f
(decl) 12 (make); make (type) 13 number (type) 14 boolean; boolean (asmt) 15 [id]; [id] (Cond) 16 if; if (Loop) 17 as; as (output) 19 say; say (ids) 20 [id] number boolean (more-ids) 21 [id] number boolean (ever) 23 FIRST, ((bool)) FIRST, ((bool))		10	as		as
(type) 13 number ; number ; hoolean (type) 14 boolean ; boolean (asmt) 15 [id] ; [id] ; [id] (Cond) 16 if ; if (Loop) 17 as ; as (output) 18 tell ; tell ; tell ; say ; say ; say ; say (ids) 20 [id] number boolean ; [id] (more-ids) 21 [id] number boolean ; fid] (more-ids) 22 & ; first, (bool) ; First, (bool)		11	tell say	,	tell Say
(type) 14 boolean boolean (asmt) 15 [id] ; [id] (Cond) 16 if ; if (Loop) 17 as ; as (output) 18 tell ; tell (q say say say (ids) 20 [id] number boolean [id] (more-ids) 21 [id] number boolean ; (ever) 23 FIRST, ((bool)) . FIRST, ((bool))	(decl)	12	(make)	;	make
(asmt) (5 [id] ; [id] (Cond) (6 if ; if (Loop) 17 as ; as (output) (9 say ; say (ids) 20 [id] number boolean [id] (more-ids) 21 [id] number boolean ; solean ; and say (ran) 23 First, (bool)	(type)	13	number		number
(Cond) 16 if ; if (Loop) 17 as ; as (output) 18 tell . tell (output) 19 say ; say (7ds) 20 [7d] number boolean [7d] (more-ids) 21 [7d] number boolean frumber boolean (23 FIRST, ((bool)) . FIRST, ((bool))		14	boolean	,	boolean.
(loop) 17 as; as (output) 18 tell tell (output) 19 say (rds) 20 [rd] number boolean (more-ids) 21 [rd] number boolean 22 E; FIRST, ((bool)) (ever) 23 FIRST, ((bool))	(asmt)	15	[7d]	j	[id]
(output) 18 tell tell (output) 19 say (7ds) 20 [7d] number boolean (more-ids) 21 [7d] number boolean (more-ids) 22 E (ever) 23 FIRST, ((bool)) (ever) (state of the state of the stat	(Cond)	16	if	;	if
(output) 19 say (7ds) 20 [7d] number boolean [7d] (more-ids) 21 [7d] number boolean 7umber boolean FIRST, ((bool))	(200p)	17	٩S	;	as as
(7ds) 20 [7d] number boolean [7d] (more-ids) 21 [7d] number boolean field] (23 FIRST, ((bool)) FIRST, ((bool))	(output)		tell		tell
(more-ids) 21 [id] number boolean [id] (more-ids) 22 & inumber boolean (ever) 23 First, ((bool)) . First, ((bool))		19	Say	. ,	Say
(more-ids) 22 E; FIRST, ((bool)) (ever) 23 FIRST, ((bool)) FIRST, ((bool))	<7ds>	20	([7d]	number boolean	[rd]
(ever) 23 FIRST, ((bool)) FIRST, ((bool))	(More-185)	21	[7d]	number boolean	£2 61
		22	<u>ع</u>	j	triumber boolean
24 FIRST, ((arith)) FIRST, ((arith))	(000	23	FIRST, ((bool)		
	17NY/	24	FIRST, ((arith))	,	FIRST, (Karith)

A3. LL(1) Parse Table (Continued)

Production	ns	FIRST,	FOLLOW,	LOOK AHEAD,
	25	fact		fact
(1	26	lie	7	lze
(bool)	zγ	not	,	mot
·	28	less is more		less is more
	29	and		and
(bool-tail)	30	٥r	j	or
	31	ક		;
	3z	less		less_
(test)	33	īs	[id] [const] (is
	34	more		more
	35	[Idj	C 12 Fam. 47	C7d1
(arith)	36	[const]	; (id) [const]	[const]
	37	((?)	(
	38	plus	· C2-12-C2 -+2	plus
(arith-tail)	39	times	; [1d] [cont]	times
	40	3		; Cidj [const1

A4. Deterministic LL(1) Parser

