BNF Assignment

Learning Abstract

This assignment is all about BNF. I have composed some BNF grammars for given languages and drawn some BNF parse trees alongside describing BNF in English, in a straightforward, compelling manner.

Problem 1: Shapes

BNF Language:

<shapes> ::= (<size-spec> <color-spec> <pattern-spec> <shape-spec>)
<size-spec> ::= (size <size-val>)
<color-spec> ::= (color <color-val>)
<pattern-spec> ::= (pattern <pattern-val>)
<shape-spec> ::= (shape <shape-val>)
<size-val> ::= large | medium | small
<color-val> ::= red | blue | yellow
<pattern-val> ::= striped | dotted | solid
<shape-val> ::= circle | square | triangle



((size big)(color blue)(pattern solid)(shape circle))





Task 2: Special Quaternary Numbers(SQN)

BNF Language:

Parse Trees:

0. (Lsqn) Town 132.00 (sqn) (Shot - Zero) (not-one) 1- rei rad (not-three) 3 (not - two) (empty) 1223 (< sqn) Cutaris (not-zero) not-one) (Inot -two) 2 Here, since the definition of the non-terminal (not two) does not allow a 2, a passe tree cannot be dravon for 1223.

Task 3: Fours

BNF Language:

<fours></fours>	::= <one-list> <two-list> <four-list></four-list></two-list></one-list>
<one-list></one-list>	::= <empty> (1 1 1 1) <one-list></one-list></empty>
<two-list></two-list>	::= <empty> (1 1 2) <two-list> (1 2 1) <two-list> (2 1 1) <two-list> </two-list></two-list></two-list></empty>
	$(2 2) \leq \text{two-list} >$
<three-list></three-list>	$::= \langle empty \rangle (1 3) \langle three-list \rangle (3 1) \langle three-list \rangle$
<four-list></four-list>	$::= \leq empty > (4) \leq four-list >$

Parse Trees:

	Birt-100)
2.	(4)
1	
1	(Danie)
CHIGAN	(founds)
((Lone-list) (two-list) (three-list) (powr-list)
-	(empty) (empty) (empty) ///
	Chot-2010
	((3(4)))
	() (Snot-ones)
	(Lever)
1	(agit tars) (s)
2	(Lempty)
1020 1	HERE SINCE FILE
will be	She Sheer of the
tonss	the non-teaming
00	Aces not allow -



Task 4: BXR

```
<BXR> ::= <sequence> | <value>
<sequence> ::= <operation> | <value><sequence> | <sequence><value> |
<sequence><sequence> | <value> | <empty>
<operation> ::= <and> | <or> | <not>
<and> ::= (and <sequence><sequence>)
<or> ::= (or <sequence><sequence>)
<not> ::= (not <value>)
<value> ::= #t | #f
```

	(or #+) (++ (++ +++) +++)
	(KBXR)
	(sequence)
	(coperation)
	Gord
	() (or) (sequence) (sequence) ()
	(value) (empty)
-	() (HE)



Task 5 : CF (Color Fun)

```
<cf> ::= <add> | <colors> | <show> | <describe> | <exit>
<add> ::= add (<n><n><n>) <color> | add (<n><n><n><n><n>) <color>
<show> ::= show<color>
<colors> ::= <color><colors> | <empty>
<describe> ::= describe <color>
<exit> ::= Goodbye...
<n> ::= 0|1|2|3|...255
<color> ::= purple | red | light-red | favorite-color | <empty>
```

	- Maria
celons rolas-stward (06) act 0011160	
	- the
((+2))	
(Lidous)	A State
	100
Eldard Crokel	
Court Colors	ALC: NO
(contact) and all	
propres (conor) (conors)	
(TA) (LIDE) (LIDE)	
(great (2 collor)) (color)	
	-
(light-sted) (COLOS) (COLOS)	and -
	-
(favorite-) (Color) Kiele	Pro)
· color	
(Compty) Cerno	tur
	4V

< 8 5) 11 E Kolders show pumple 660 = 11 < 660 (Lef) Letters / Cempty) (C show) (color) (show) color squap ple (set [light set] favoint add (100 120 170) favouite - color 10/00 (LCF) Kadd) (A) (color) (A) favourte-color (n) add (120) (100) (bare) Right sie

Task 6: BNF?

BNF (the abbreviation for Backus Naur Form) is a form of context-free grammar which concisely defines the rules for the creation of a language. It sets the fundamentals for the language based on which expressions in the language can be created. The core elements of BNF include:

- Tokens, which are the part of the language being defined
- Non-terminals, which support the language being defined
- Productions, which map a non-terminal to a string of tokens and non-terminals, and
- A start symbol