Grammar #1: Hexadecimal Arithmetic

First Grammar for Signed Addition and Subtraction of Hexadecimals

Starting with non-terminal (her_literal)

(her_literal) ::= 0x (number)

(number) ::= (number)(number) | digit

(digit) ::= 0 | 1| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter> ::= A | B | C | D | F

The above grammar describes the format of hexadecimal literals in the Java languages. String 0XF3 is an element of the language generated by this grammar.

A hex is a base sixteen (x_{16}) number/ value written as <hex> ::= <digit> | <letter> | <hex> <hex>

Given Hexadecimal numbers **CABB$_{16}$** and **BBAC$_{16}$** we will perform the two operations; Addition and Subtraction as said in the assignment

Binary	**Hex**	**Decimal**
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11

1100	C	12
1101	D	13
1110	E	14
1111	F	15

Distinct derivation sequences existing for this string

In binary form **$CABB_{16}$** is written as 1100101010111011_{16} while **$BBAC_{16}$** is written as 1011101110101100_{16}

Addition

For my case I have chosen letters for the hexadecimals

(Addition) :: = {<CABB>} <+> {<BBAC>}

(CABB) ::=<C> <A> <B> <B>

(C) ::= 1100

(A) ::= 1010

(B)::= 1011

(B) ::= 1011

(BBAC) ::=<B> <B> <A> <C>

(B) ::= 1011

(B) ::= 1011

(A) ::= 1010

(C) ::= 1100

CABB + BBAC =**18667**

Subtraction

Also, For my case I have chosen letters for the hexadecimals

(Subtraction) :: = {<CABB>} <-> {<BBAC>}

(CABB) ::=<C> <A> <B> <B>

(C) ::= 1100

(A) ::= 1010

(B)::= 1011

(B) ::= 1011

(BBAC) ::=<B> <B> <A> <C>

(B) ::= 1011

(B) ::= 1011

(A) ::= 1010

(C) ::= 1100

CABB-BBAC =**F0F**

Grammar #2: A Mini Programming Language

miniature programming language that supports the following:

1. Creating a variable using the following format (VAR keyword followed by a variable name)

    VAR name
    Note: variable names may contain upper- and lower-case letters and numbers and must begin with a letter of either case

I will choose a "MiniC." Grammar

Being a simple subset of the standard C language, ***MiniC*** creates a variable using VAR keyword followed by a variable name. It does not include arrays, structs, unions, files, sets, switch statements, do statements, or many of the low level operators.

Function	->	Type identifier (ArgList) CompoundStmt
ArgList	- >	Arg | ArgList , Arg
Arg	->	Type identifier
Declaration	->	Type IdentList ;
Type	->	int | float
Compare	->	== | < | > | <= | >= | !=
Factor	->	(Expr) | | identifier | number

Example

int Y=40;

float d=2.2

2. **Defines a signed (decimal) integer data type that consists of an optional sign specifier and any number of digits 0-9**

1, -127

For 1

<integer> b 1<digit>

B1

For -127

float::= “.” Digits “.” [digits]

digits::=digit {digit}

digit::=”-1”|”2”|”7”

3. **Defines a string data type that can hold an arbitrary-length string consisting of upper- and lower-case letters, spaces, and periods, and is contained within quotation marks:**

"Corrputer Organization" "Another valid string .”

String::= <s>

String::=<a>

s::= “Corrputer Organization”

a::= “Another valid string .”

3. **An assignment operator that assigns the value of one variable to another variable, an integer value to a variable, a string to a variable, or an expression (given in next step) to a variable:**

`x = 0, y = x, s= "a string", x = y + z`

int x,y,z;

String s;

S::= "a string"

Print::= “enter the values of the variable names”

X::=0;

Y::=x;

X::=y+z;

4. **The following 5 binary operations: addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (**) of either two variables, two literal values, or one variable and one literal value**

x + y, 10 + 20, 2 * pi, "Hello + "world"

Note: your grammar should support string addition (concatenation), but all other operators are for the signed decimal integer type created in Step #2.

int x,y;

String S;

Print::= “enter values of x and y”

Scan the values entered

a) Addition

X::=10

Y::=20

X+y::=10+20

30

b) Subtraction

X::=10

Y::=20

X-y::=10+20

-10

c) Multiplication

X::=10

Y::=20

Use pi

Answer::= pi*(x*y)

d) Exponentiation

C=4e6

x=lo(2)/30

T=45

Y=c*exp(k*t)

6. A print statement that prints the contents of a variable and has the following format:

PRINT where VAR is the 'name' value from a variable defined in Step #1

```
#include <stdio.h>
Int main()
{
int b,c,d;
printf(“Enter value1”);
scan the value1

print (“enter value2”)
Scan the value2

d=b+c;

printf(“%d +%d=%d\n, b,c,d)
return 0;
}
```