1. Regular Expressions (30 points)

Consider the following regular expression which has been divided into three parts, 1,2,3.

```
------1--------    -------2------     -------3-------
   (x | y)*          (x | z)            (w*  | z*)+
```

(5 points each) For each string below, either write that it is not generated by the regular expression (i.e., **NOT GENERATED**) or **circle and label** the sections of each string generated by the regular expression parts **1,2,3**. The following example shows what we mean.

```
   x     x      x      z     w
-------------1---------  --2--  --3—
```

a) y x w z

b) y y y y y y z

c) x z w x

d) x x x x x x x w

e) x z x z z w

f) x z z z z z

2. Grammars, Ambiguity, Precedence (60 points)
Below is a grammar with two operators:

S → E
E → - E | E + E | id

a) (10 points) This grammar is ambiguous. Prove that the grammar is ambiguous.

b) (5 points) What is the precedence of unary - wrt +? (choose one)

- higher precedence than +_______________

- equal precedence to +_______________

- lower precedence than +_______________

c) (10 points)** Give evidence to support your answer to part b.

d) (5 points) What is the associativity of the operator + in the grammar in part a)? (check one)

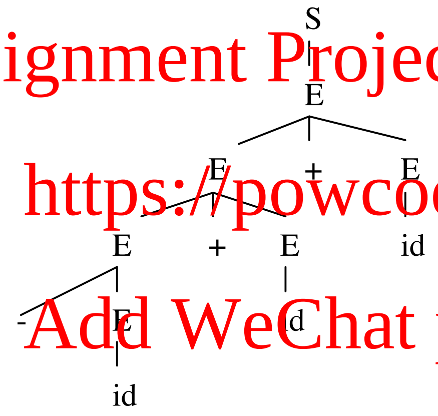
Right associative______________,

Left associative______________

Both left and right associative__________________

e) (10 points)** Give evidence supporting your answer to d).

Suppose we wanted expression **-id+id+id** to have only one parse tree (call this property P):



f) (10 points)** Describe in English the precedence and associativity rules necessary to ensure property P.

g) (10 points)** Modify the original grammar so it is not ambiguous, and it meets the constraints of property P.

Name: ___________________________

3. LL Parsing (60 points)

Consider the following grammar over terminals ***{c,d,e}***. ***S*** is the starting symbol of the grammar.

S → TS | [S]S | ε
T → (X)
X → TX | [X]X | ε

a) **(30pts)**** Fill in the table below with the FIRST and FOLLOW sets for the nonterminals in this grammar:

	FIRST	FOLLOW
S		
T		
X		

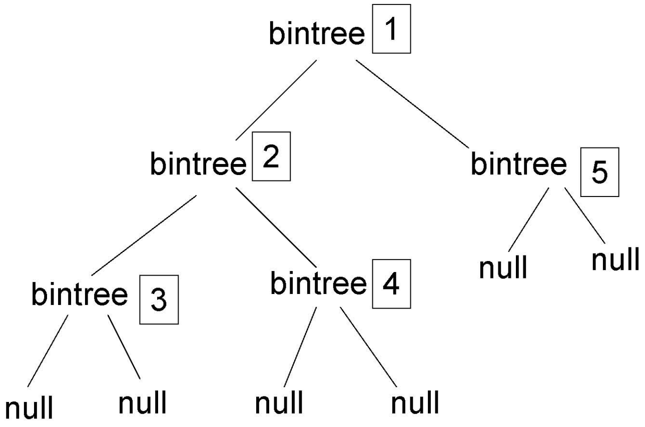
b) **(20 points)**** Fill in the column headings and the row corresponding to A in the LL(1) parsing table for this grammar:

	(	[	]	)	$
S					
T					
X					

c) **(10 points)** Is this grammar LL(1)? Explain briefly why or why not.

5. Prolog (30 points)**

Assume we are building binary trees in Prolog like the one shown below, using the following conventions:
--Each internal node has an integer label (shown in a box below) and less than or equal to 2 child nodes
--Each internal node is referred to by the functor **bintree**
--Each leaf node is represented by a Prolog literal **null.**



bintree(1, bintree(2, bintree(3,null,null), bintree(4,null,null)), bintree(5,null,null))

Write the Prolog clauses for the **walk** predicate which performs a preorder traversal of such binary trees and returns a list of the node labels encountered in preorder.

Name: ___________________________

Part 1. Scoping

Problem 1 (12 points).

Below is a program in a Pascal-like language. The language uses **static (i.e., lexical) scoping** for the lookup of non-local variables and non-local routines.

```
x : integer := 0

procedure A(n : integer)
   if n < 2
     x := x + 1
     B(n+1)
   else
     write("A: ", x)

procedure B(m : integer)
   x : integer := 100
   write("B: ", m)
   A(m)

/* begin of main */
A(0)
/* end of main */
```

a) (4 points) Show the output of the execution.

b) (4 points) Show the frames on the stack when `write("A: ", x)` gets called. For each frame, show the static (i.e., lexical) and dynamic links. (Use the drawing to the right.)

c) (4 points) Now, suppose the language used dynamic scoping. What would the output be?

procedure
___main_____
lexical-link xxxx
dynamic-link xxxx

procedure _________
lexical-link
dynamic-link

procedure _________
lexical-link
dynamic-link

procedure _________
lexical-link
dynamic-link

procedure _________
lexical-link
dynamic-link

procedure _________
lexical-link
dynamic-link