

CS 160: Practice Sheet 3 (do not turn in)

#1 For the given attribute grammar, answer the following questions:

- The following grammar uses inherited attributes? T/F
- The following grammar uses synthesized attributes? T/F
- The following grammar is an L-attributed grammar? T/F
- The following grammar is an S-attributed grammar? T/F
- What is $A.n$ for the following sentence “**a a x a a a y a**”?
- What does the attribute grammar below compute?

$$\begin{array}{lll}
 A & \rightarrow & B y C \quad \{A.n \leftarrow B.n + C.n\} \\
 B_0 & \rightarrow & \mathbf{a} B_1 \mathbf{a} \quad \{B_0.n \leftarrow B_1.n + 2\} \\
 & | & \mathbf{x} \quad \{B.n \leftarrow 0\} \\
 C & \rightarrow & \epsilon \quad \{C.n \leftarrow 0\} \\
 & | & \mathbf{a} \quad \{C.n \leftarrow 1\}
 \end{array}$$

#2 Disk Arm Checking with Attributes

Consider the following grammar, which defines a simple language that describes movements of the arm of a disk drive (Disk Arm Programs, which is abbreviated DAP). We don't want the disk arm to crash into the side of the enclosure or into the center mount, which will happen if we read from a track number that is too low or too high. The language below describes the command sequences that can be passed to the disk drive. The command **skipto** will set the track number to the value of *tracknum* (denoted *tracknum.num*). You can assume that *tracknum.num* is given. The command **track_left** will decrement the track number by one, and **track_right** will increment the track number by one. You can assume that the disk arm starts at track 1. For example, the program (command sequence): **start skipto 5 track left skipto 5 track left track left** will leave the disk head at track 3.

$$\begin{array}{lll}
 S & \rightarrow & DAP \quad \{ \quad \} \\
 \\
 DAP & \rightarrow & DAP Dir \quad \{ \quad \} \\
 \\
 & | & DAP \mathbf{skipto} tracknum \quad \{ \quad \} \\
 \\
 & | & \mathbf{start} \quad \{ \quad \} \\
 \\
 Dir & \rightarrow & \mathbf{track left} \quad \{ \quad \} \\
 \\
 & | & \mathbf{track right} \quad \{ \quad \}
 \end{array}$$

a) Add an attribute called *pos* to the above grammar. When *S.pos* is calculated, it should be the final position of the disk arm when the command sequence is done (for the example program above, *S.pos* should be 3).

b) What sort of attribute is *pos* (inherited, synthesized)?

c) Draw out the parse tree for the sentence **start track left track right skipto 7 track right** and draw the value of the attributes at each node in the tree (decorate the parse tree)

#3 Procedures

- Show the contents of a lexically scoped symbol table for the program on the right (the language is similar to C but with the addition of nested procedures).
- Draw the activation tree for the following execution sequence: main calls p2, p2 returns, main calls p1, p1 calls p2, p2 calls p3, p3 calls p3, p3 returns, p2 calls p1, p1 returns, p2 returns, p1 returns.
- In (b) above, what (which activation records) will be on the stack after the second call to p3?

```
procedure main()
{
    int a;
    float x, y;
    procedure p1(int a)
    {
        int b;
        float x;
        int c;
        procedure p3()
        {
            int a, c;
            ...
        }
        ...
    }
    procedure p2(float x, float z)
    {
        int b;
        ...
    }
    ...
}
```

#4 Three-Address Code

Productions	Semantic Rules
$S \rightarrow id := E$	$id.place \leftarrow lookup(id.name);$ $S.code \leftarrow E.code \parallel gen(id.place := E.place);$
$E \rightarrow E1 + E2$	$E.place \leftarrow newtemp();$ $E.code \leftarrow E1.code \parallel E2.code \parallel gen(E.place := E1.place + E2.place);$
$E \rightarrow E1 * E2$	$E.place \leftarrow newtemp();$ $E.code \leftarrow E1.code \parallel E2.code \parallel gen(E.place := E1.place * E2.place);$
$E \rightarrow (E1)$	$E.code \leftarrow E1.code;$ $E.place \leftarrow E1.place;$
$E \rightarrow - E1$	$E.place \leftarrow newtemp();$ $E.code \leftarrow E1.code \parallel gen(E.place := 'uminus' E1.place);$
$E \rightarrow id$	$E.place \leftarrow lookup(id.name);$ $E.code \leftarrow ''$ (empty string)

Consider the above productions and semantic rules, along with the following program fragment: $x := (a+b) * (c+a)$

- Write the three-address code that would be generated from the semantic rules shown for the program fragment.
- How many temporary variables are used to implement this?
- The programming language designer has decided to add the pre-increment operator “++” to the grammar with the production “ $E \rightarrow ++ id$ ”. This new operator increments id , and then returns the value (for example, after the code, $x=5; y=++x;$ is executed, x is 6 and y is 6). Add the semantic rule that will generate code for this new operator.

#5 Arrays

0,0	0,1
1,0	1,1
2,0	2,1

a) Given the array (above) that is in two dimensions, show the layout of the array if row-major order is used. (Use the squares below)

--	--	--	--	--	--

b) If the user has specified the array above, what would be the formula for getting element $A[x,y]$ in the array? Use actual numbers for the bounds of the array in the formula and simplify as far as possible.