Homework # 2
due Monday, September 18, 2:00 PM

1 Ambiguous Grammars

Some programming languages have "if" expressions:

```
<expr> ::= <expr> + <expr> | <expr> = <expr> | <expr> * <expr>
        |  if <expr> then <expr> else <expr>
        |  <INTEGER> | <ID> | ( <expr> )
```

Thus one can write:

```
(if x = 0 then 1 else 10 * x ) + y
```

However the grammar above is ambiguous.

- Show all five parse trees for the string "`if 1 = 2 then 3 else 4 * 5 + 6`" For each one, indicate what the computed result would be.
- In the next step, I'll ask you to write an unambiguous grammar that accepts the same language. But doing so is tricky. So first, find what's wrong with the following (three) "solutions." For each one, indicate if there are *false negatives* (strings that are *not* accepted, even though they should be), *false positives* (strings that are accepted, even though they should *not* be) or *ambiguous parses* (strings for which there is more than one parse tree). You must give concrete examples. (Concrete examples have no nonterminals in them.)

  1.
     ```
     <expr> ::= if <expr1> then <expr1> else <expr1>
     <expr1> ::= <expr2> = <expr2> | <expr2>
     <expr2> ::= <expr3> + <expr3> | <expr3>
     <expr3> ::= <expr4> * <expr4> | <expr4>
     <expr4> ::= <ID> | <INTEGER> | ( <expr> )
     ```

  2.
     ```
     <expr> ::= if <expr> then <expr> else <expr> | <expr1>
     <expr1> ::= <expr1> = <expr1> | <expr2>
     <expr2> ::= <expr2> + <expr2> | <expr3>
     <expr3> ::= <expr3> * <expr3> | <expr4>
     <expr4> ::= <ID> | <INTEGER> | ( <expr> )
     ```

  3.
     ```
     <expr> ::= if <expr1> then <expr1> | <expr1>
     <expr1> ::= <expr2> = <expr1>
              |  <expr1> + <expr2>
              |  <expr2> * <expr2>
              |  <expr1> else <expr1>
     <expr2> ::= <ID> | <INTEGER> | ( <expr> )
     ```

- Now rewrite the grammar to be unambiguous, where: `if` has lowest precedence, then `=`, then `+` and then `*`, with highest precedence. Both `+` and `*` should have left associativity, and `=` should be non-associative (`x = y = z` should be a syntax error). Other than the last requirement, the grammar should have no false negatives, false positives or ambiguous parses.

2 Standard compilation sequence

Write a file `square.cc`:

```
#include <iostream>
using namespace std;

int main()
{
  int i;
  cin >> i;
  cout << i*i;
  exit(0);
}
```

On `andrew.cs.uwm.edu` (ssh using port 53211), first compile this program to assembly (`g++ -S square.cc`) and then to an object file (`g++ -c square.cc`) and finally to an executable (`g++ square.cc`). Answer the following questions:

1. Look in the assembly file (`square.s`) and find out in the function `main` how it refers to `cin`. Show that here. Run `c++filt` and then type a "mangled" name to show a readable name. What is the full name of `cin` ? Why does C++ mangle the name?
2. `main` is a "`@function`" label. What other function labels are defined in the assembly file? Give their unmangled names.
3. Use `readelf -a square.o` to examine the object file, `square.o`. How does the file refer to `exit` ?
4. Look at the relocation table for the text segment: `.rela.text`. What does `_ZNSolsEi` refer to? Why is it there? Why is there *no* entry for `int i;` ?
5. Use the command `ldd` on the executable `a.out`. This indicates what shared libraries are used by the executable and where they (currently) can be found. List each and investigate what each does; report why each is being used.

3 Submitting Your Work

Turn in your answers on paper.