CMSC 430 – Example of quad production 10/23/01

Original Program:

```
int A(int X)
{X = 12345;
      return X;}
void B
{int X, Y;
      Y = 0;
      while (25>Y)
            {Y = Y+1;}
      printStr("Y=");
      printInt(Y);
      X = A(Y);
      printInt(X);
      println();}
```

Symbol table at end of parsing:

ST Locn	NAME	Type
1	A	Procedure
2	X	Integer parm
3	B	Procedure
4	X	Integer
5	Y	Integer
6	T1	Label
7	T2	Integer temp
8	T3	Label
9	T4	Integer temp
10	L1	String literal

Program with quad output:
Note: If the virtual machine executed these quads, then compilation would be completed. However, since the machine doesn't, the code generator must convert each quad into a sequence of instructions that has the same effect:

```
int A (int X)
     <NEWPROC, 1, 0, 1>        ⇐Start of A
     <PARAM, 2, 0, 0>          ⇐parameter X
     <STARTPROC, 0, 0, 0>      ⇐decl part of A
{X = 12345;
     <=, 12345, 0, 2>          ⇐Asgn of 12345 to X
                               ⇐Note: Need to differentiate
                               integer 12345 and symbol table
```

address 2

```
return X;}
      <RETURN,  0, 0, 2>       ⇐Return of value X
      <ENDPROC, 0, 0, 0>       ⇐End of procedure A
void B
     <NEWPROC, 3, 0, 0>        ⇐Start of procedure B
     <STARTPROC, 0, 0, 0>      ⇐Start of Decl part of B
{INT X,                        ⇐ No quads - only symbol table
    Y;
Y = 0;
     <=, 0, 0, 5>              ⇐ Asgn of 0 to Y
while (25>Y)
     <LABEL, 6, 0, 0>          ⇐Loop back to top of while
     <>, 25, 5, 7>             ⇐ Computer 25>Y. Save in T2
     <IFTEST, 7, 8, 0>         ⇐Test while expr and jump to T3 (8)
                                  if false
{Y = Y+1;
     <+, 5, 1, 9>              ⇐ Y+1 stored in T4
     <=, 9, 0, 5>              ⇐ Y+1 stored in Y
}
     <GOTO, 6, 0, 0>           ⇐Jump to top of while stmt
     <LABEL, 8, 0, 0>          ⇐Jump here if iftest is false
printStr("Y=");
     <printstr, 10, 0, 0>      ⇐ Write literal
printInt(Y);
     <printint, 4, 0, 0>       ⇐Write Y
X = A(Y);
     <ARG, 5, 0, 0>            ⇐Argument Y to procedure
     <CALL, 1, 1, 0>           ⇐Call function 1 (A) with 1 argument
     <=, -1, 0, 2>             ⇐Save fcn value (-1) in X
printInt(X);
     <printInt, 2, 0, 0>       ⇐Write Y
println();}
     <println, 0, 0, 0>        ⇐Write new-line
     <ENDPROC, 0, 0, 0>        ⇐End of procedure B
```