CS 316 9-10-09

1. Diagram:
   (a) Parse Tree
      i. Semantic Analysis / Intermediate Code Generation
         A. Semantic Errors
         B. Intermediate Code –
            {{Intermediate-code compilers stop here.
            They're executed by virtual machines}}
            - Optimization –
              {{Improve execution speed.
              Reduce code size (actual number of lines)}}
              - Optimized intermediate code
                - Object Code Generation
                  - Object Code –
                    {{almost executable code.
                    Relative addresses.
                    This is the end of native-code compilers}}
                    - Linking/Loading
                      {{Library functions code
                      Analysis phase (front end)
                      Lexical
                      Syntax
                      Type
                      Semantic
                      OS assigns actual physical memory address
                      Generation phase (back end)
                      Intermediate code optimization
                      Object code}}
                    - Executable machine code in memory
      ii. Type Checking – Check type consistency and type mismatches
         A. Type Errors
         B. Type-Checked Parse Tree (points back up to Semantic Analysis / Intermediate Code Generation)
   (b) Examples of semantic analysis:
      i. Variable Declaration – All variables must be declared (with types) at suitable places
      ii. Function Declaration – Can't call undeclared functions
      iii. Scope Checking
      iv. Can't declare the same variable/function name twice in the same scope
      v. Calling functions with incorrect number of parameters
      vi. Source-code-level optimization
         A. Example: inlining of function calls (eliminating function calls by substituting the function body)
            - Inlining example:

```
void f()
{
        B;
}//f
```

```
int main()
{
.
        f();            //inlining would eliminate this function call and put “B;” in
it's place.
.
}//main
```

2. Formal Description Of Lexical Items And Syntax
   (a) A BNF (Backus-Naur Form) grammar (i.e. context-free grammar) is a triple (T,N,R)
      i. T is a set of terminal symbols – ASCII/Unicode characters
         A. $T=\{0,1,\ldots,9,a,b,\ldots,z,A,B,\ldots,Z,+,-,*,/,(,),.\}$
      ii. N is a set of nonterminal symbols – Syntactic categories <X> where X is a mnemonic identifier
         A. N = {<digit>, <letter>, <int>, <id>, <rest>, <float>, <exponent>, <eSign>, <sign>, <E>}
            - <rest> helps define <id>
            - <exponent>, <eSign>, and <sign> are for scientific notation of floats
      iii. R is a set of production rules $\langle X\rangle\to\alpha_1|\alpha_2|\ldots|\alpha_n| where\, n\geqslant 1$

$$\alpha_i is\, any\, string\, of\, terminals \wedge ¿ \vee nonterminals, \vee can\, be\, the\, empty\, string\, \in$$

         A.

$$R=\{\langle digit\rangle\to ٠|١|٢|٣|\ldots|٩\}$$

$$\{\langle letter\rangle\to a|b|\ldots|z|A|B|\ldots|Z\}$$

$$\{\langle integer\rangle\to\langle digit\rangle|\langle digit\rangle\langle integer\rangle\}$$

$$\langle id\rangle\to\langle letter\rangle\langle rest\rangle$$

$$\langle rest\rangle\to\in|\langle letter\rangle\langle rest\rangle|\langle digit\rangle\langle rest\rangle$$

      iv. Derivations – Let $\beta\langle X\rangle\gamma$ be any string with an occurrence of <X>.
         β,γ are any strings of terminals/nonterminals, including $\in$.
         Let $\langle X\rangle\to\alpha_i$ be one choice of a production rule for <X>.
         Then $\beta\langle X\rangle\gamma$ is said to derive $\beta\alpha_I\gamma$ in one step, denoted $\beta\langle X\rangle\gamma\Rightarrow\beta\alpha_i\gamma$
         A. Example: Derive “316” from <int>
            $\langle integer\rangle\Rightarrow\langle\underline{digit}\rangle\langle integer\rangle\Rightarrow 3\langle\underline{integer}\rangle 3\langle\underline{digit}\rangle\langle integer\rangle\Rightarrow 31\langle\underline{integer}\rangle\Rightarrow 31\langle\underline{digit}\rangle\Rightarrow 316$
         B. Example: Derive “A2C3” from <id>
            $\langle id\rangle\Rightarrow\langle\underline{letter}\rangle\langle rest\rangle\Rightarrow A\langle\underline{rest}\rangle\Rightarrow A\langle\underline{digit}\rangle\langle rest\rangle\Rightarrow A2\langle\underline{rest}\rangle\Rightarrow A2\langle\underline{letter}\rangle\langle rest\rangle\Rightarrow A2C\langle\underline{rest}\rangle\Rightarrow$
      v. For any syntactic category <X>, the language defined by <X> is defined as:
         A. $\{\beta|\langle X\rangle\overset{asterisk}{\Rightarrow}\beta,\beta\, is\, a\, string\, of\, \underline{terminals}\}$
            $where\,\gamma_1\overset{asterisk}{\Rightarrow}\gamma_2 means\,\gamma_1 derives\,\gamma_2\in zero\vee more\, steps.$
            $The\, language\, defined\, by\,\langle integer\rangle is:$
            $\{\beta|\langle integer\rangle\overset{asterisk}{\Rightarrow}\beta,\beta\, is\, a\, string\, of\, terminals\}$