BugsWorld Introduction
BugsWorld

• You will be working on projects related to various aspects of a game called *BugsWorld*:
  – The Game
  – The Simulator
  – The Language
  – The Compiler
The Game: It’s a Bug’s World!
The Simulator

• The BugsWorld game is played on a system consisting of:
  – Server
  – Clients
  – Displays
The Simulator

• The BugsWorld game is played on a system consisting of:
  – **Server**

The server keeps track of “the world”, processes client requests, resolves conflicts.
The Simulator

- The BugsWorld game is played on a system consisting of:
  - Server
  - Clients

Each client program simulates creature (bug) behavior for all creatures of one species.
The Simulator

• The BugsWorld game is played on a system consisting of:
  – Server
  – Clients
  – Displays

Each display shows the current state of the world, plus some statistics about the simulation.
The Simulator

- The BugsWorld game is played on a system consisting of:
  - Server
  - Clients
  - Displays

Each process can run on a different computer (distributed simulation).
The Language

• The behavior of each species is determined by a program in the language **BL**
• Primitive instructions: *move, turnleft, turnright, infect, skip*
• Control structures: *IF-THEN, IF-THEN-ELSE, WHILE-DO*
• Defining new instructions: *INSTRUCTION-IS*
• Conditions: test whether “next” cell is *empty, friend, enemy, or wall* (plus *true and random*)
PROGRAM TryToGuess IS

INSTRUCTION FindObstacle IS
    WHILE next-is-empty DO
        move
    END WHILE
END FindObstacle

...
PROGRAM TryToGuess IS

INSTRUCTION FindObstacle IS
  WHILE next-is-empty DO
    move
  END WHILE
END FindObstacle

...
BEGIN
   WHILE true DO
      FindObstacle
      IF next-is-enemy THEN
         infect
      ELSE
         IF next-is-wall THEN
            turnleft
         ELSE
            skip
         END IF
      END IF
   END WHILE
END TryToGuess
BL Features

• Precise syntax
• Case sensitive
• Matching ENDS
• Identifiers:
  – Start with any of 'a'..'z', 'A'..'Z'
  – Followed by any of 'a'..'z', 'A'..'Z', '0'..'9', '-'
PROGRAM TryToGuess IS
  INSTRUCTION FindObstacle IS
    WHILE next-is-empty DO
      move
    END WHILE
  END FindObstacle
BEGIN
  WHILE true DO
    FindObstacle
    IF next-is-enemy THEN
      infect
    ELSE
      IF next-is-wall THEN
        turnleft
      ELSE
        skip
      END IF
    END IF
  END WHILE
END TryToGuess

<20, 15, 20, 6, 7, 0, 5, 2, 12, 12, 3, 5, 18, 8, 17, 1, 5, 18, 4, 5, 0>
PROGRAM TryToGuess IS
  INSTRUCTION FindObstacle IS
    WHILE next-is-empty DO
      move
    END WHILE
  END FindObstacle
BEGIN
  WHILE true DO
    FindObstacle
    IF next-is-enemy THEN
      infect
    ELSE
      IF next-is-wall THEN
        turnleft
      ELSE
        skip
      END IF
    END IF
  END WHILE
END TryToGuess

This string of integers is the **object code** for the BL program **source code** shown on the left.

\[
<20, 15, 20, 6, 7, 0, 5, 2, 12, 12, 3, 5, 18, 8, 17, 1, 5, 18, 4, 5, 0>
\]
Compiler Structure

Tokenizer → Parser → Code Generator

string of characters (source code) → string of tokens (“words”) → abstract program → string of integers (object code)
What You Will Do

• You will implement at least major parts of all three pieces of the BL compiler:
  – Tokenizer
  – Parser
  – Code Generator