Course Project

Lab

- F 1610-1800, BARH 254
- Lab attendance will not count towards the final grade.
- All lab sessions are "Open labs" in which you are expected to work on the course project where TA will be available for consultation.
- No other Lab assignments (other than Project steps)

Project support

- **GTA**: Buwani Manuweera, buwani.manuweera@student.montana.edu (please do not email through D2L)
- **Office hours**: T 1510-1600 and R 1100-1200, Student Success Center

- Direct all the questions related to the Project to the GTA (except help with debugging your code!)
- GTA will leave at 5pm (if no students)

Project

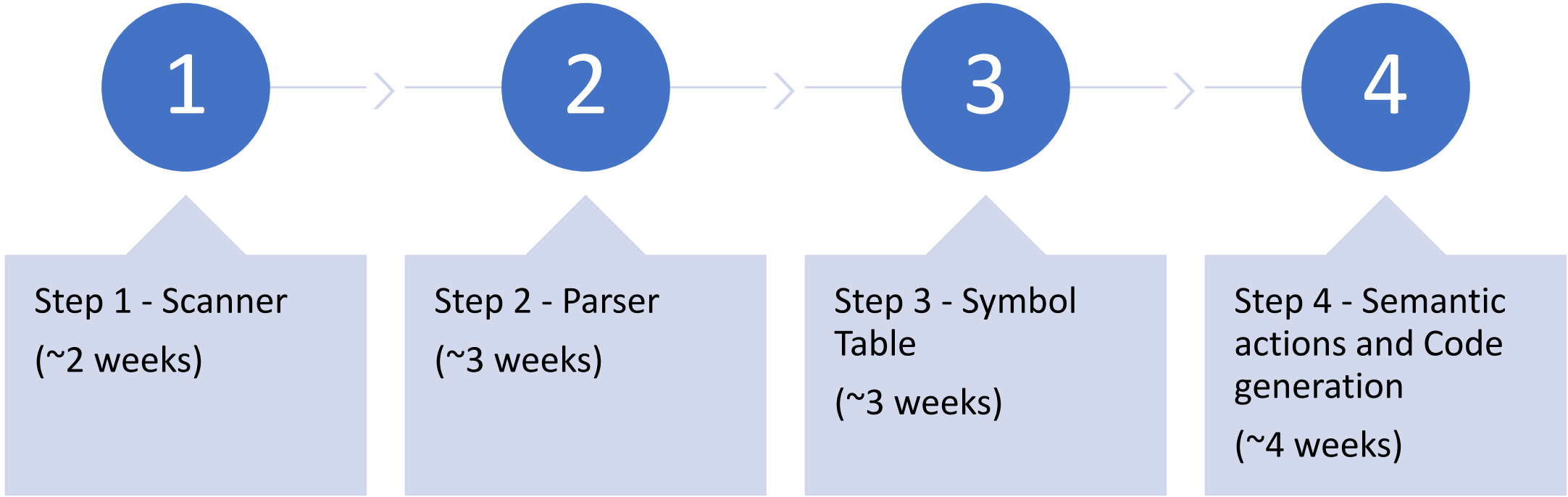
- The bulk of your final grade
- Involves implementing a full-fledged compiler for LITTLE language. See "Project" tab in D2L for more details.
- The project consists of multiple steps, each of which will be graded separately.
  - However, each step builds on the results of previous steps, so it behooves you to ensure that each step works properly.
- The majority of your project grade is based on the performance of your compiler on several predetermined test programs.

Course Grading - Project

- 50% — Project
  - implementation: 40% - 10% each step,
  - Report: 10%
  - Portfolio: Bonus points

- Report should be continuously developed.
- Portfolio is a Department Requirement.

Four steps

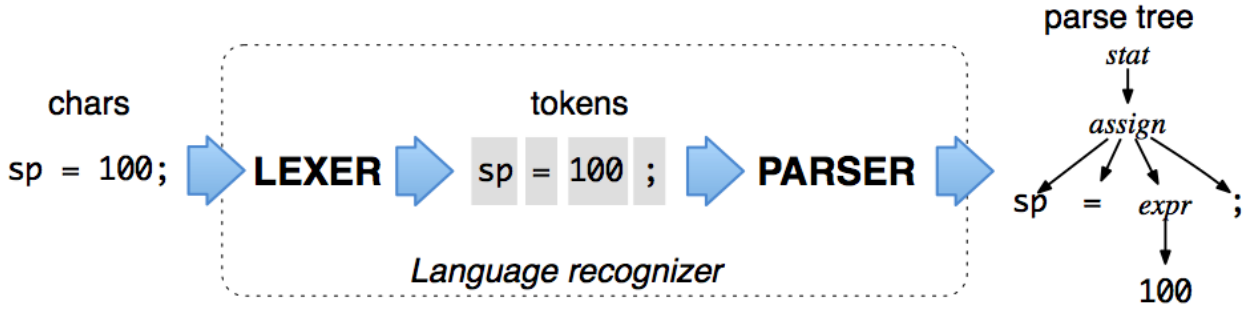

1
2
3
4
Step 1 - Scanner
(~2 weeks)
Step 2 - Parser
(~3 weeks)
Step 3 - Symbol Table
(~3 weeks)
Step 4 - Semantic actions and Code generation
(~4 weeks)

OCF (Oxiago Compiler Factory)

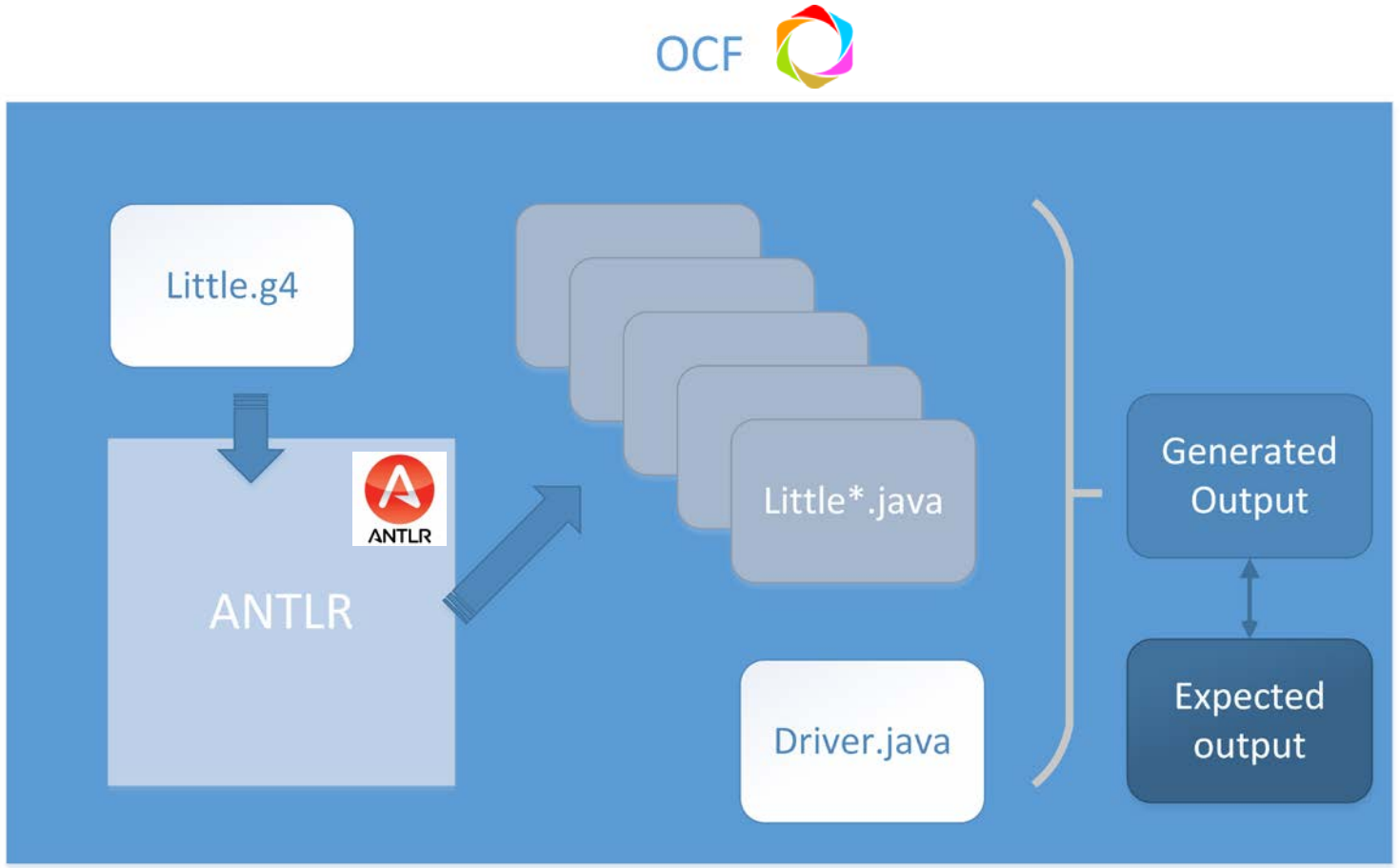
- Dedicated online compiler factory framework: http://oxiago.com/compiler/msu/
- Developed in-house by Nazmul Kazi (CS major)
- OCF uses ANTLR as its back-end.
- OCF has test cases (inputs/outputs) for each step built-in.
  - You will know how your code works on-the-fly!
- Supports Java and Python

ANTLR (Another Tool for Language Recognition)

- ANTLR is a state-of-the-art scanner/parser generation toolkit Written in Java: http://www.antlr.org
- can be used with Java and python (and some others).
- Regardless of the programming language you choose to build your compiler, GTA will only provide conceptual help regarding implementation (i.e. no technical support for debugging any code).



How OCF/ANTLR work



Project Report

- The report should detail the implementation of the compiler.
  - It should have the structure of a typical scientific paper
  - It should consist of 10-15 pages.
  - Recommended template for the project report will be given under the "Report template" tab.
  - You are expected to work together and submit just one final report per group.
  - Project report is worth 100 points.

It should be written in a progressive manner. The latest version of the report should be uploaded during each step. Failure to so may lead to deduction of points.

Project Report Template ("Project → Report")

1. Introduction
2. Background
3. Methodology (data/method/tools etc)
   a) Scanner (min. 2 pages)
   b) Parser (min. 2 pages)
   c) Symbol Table (min. 2 pages)
   d) Code generation (min. 2 pages)
4. Conclusions and Future work
5. References

E.g. at the end of step 1, the section on "Scanner" (3.a) should be complete.

Portfolio

- The portfolio is a mandatory requirement by the department (for all capstone courses).
  - It is due along with the final version of your report (at step 4 deadline).
  - Template for the portfolio will given under the "Portfolio template" tab.
  - There is no specified length. You are expected to work together and submit just one portfolio per group.

Portfolio template (“Project → Portfolio”)

- Section 1: **Program**. Attach the source listing of the program.
- Section 2: **Teamwork**. Describe how your team worked on this.
- Section 3: **Design pattern**. Identify one design pattern that was used.
- Section 4: **Technical writing**. Include the project report here.
- Section 5: **UML**. Attach the UML design diagrams for your project.
- Section 6: **Design trade-offs**.
- Section 7: **Software development life cycle model**.

Submission Instructions

- Code: there is nothing to submit. When you complete your program/code on OCF, they will be auto-graded by the system.
  - OCF uses Unix "diff -b" command to compare the outputs.
- Report:
  - use the D2L assignments folders for each step
  - E.g by the step 1 deadline: Upload your report with completed “Scanner (3.a)” section
- Portfolio:
  - use the D2L assignments folder (due: last day of the semester).

Project Discussion

- If you have questions about the project, I encourage you to post them on D2L.
- Use discussion topic: “Project”
- It’s a shared discussion forum, where your question can be answered by myself, the TA or your fellow students!
- Don’t share code!

Policies

- **(Project) Group work policy:**
  - work on the project in teams of 2 or 3.
  - Once you choose a partner(s), you must continue to work with them.
- **Late submission policy:**
  - Except for medical and family emergencies (accompanied by verification), there will be no extensions granted for project submissions. Late submissions will be scaled according to lateness as below.
    - Up to 12 hours late: 10% penalty
      12 to 24 hours late: 25% penalty
      24 to 48 hours late: 50% penalty
      More than 48 hours late: No credit

Plagiarism

- We plan to use a automatic plagiarism tracker for code
- Penalty for plagiarizing is severe
  - First time: 0 for the individual step
  - Second time: F for the course + reported to Dean's office
  - Third time: Expulsion

- Hardcoding:
  - OCF will alert/warn if it detects hardcoded outputs
  - At the first instance, your group will be flagged

Step 1

Scanner

Scanner

- A scanner's job is to convert a series of characters in an input file into a sequence of *tokens* -- the "words" in the program. So, for example, the input A := B + 4

Would translate into the following tokens:

IDENTIFIER (Value = "A")
OPERATOR (Value = ":=")
IDENTIFIER (Value = "B")
OPERATOR (Value = "+")
INTLITERAL (Value = "4")

Token Definitions (LITTLE)

```
an IDENTIFIER token will begin with a letter, and be followed by any number of letters
and numbers.

IDENTIFIERS are case sensitive.

INTLITERAL: integer number
            ex) 0, 123, 678
FLOATLITERAL: floating point number available in two different format
                yyyy.xxxxxx or .xxxxxxx
            ex) 3.141592 , .1414 , .0001 , 456.98

STRINGLITERAL: any sequence of characters except '"'
            between '"' and '"'
            ex) "Hello world!" , "***********" , "this is a string"

COMMENT:
      Starts with "--" and lasts till the end of line
      ex) -- this is a comment
      ex) -- anything after the "--" is ignored

Keywords

PROGRAM,BEGIN,END,FUNCTION,READ,WRITE,IF,ELSE,FI,FOR,ROF,RETURN,INT,VOID,STRING,FLOAT,
WHILE, ENDIF, ENDWHILE

Operators
:= + - * / = != < > ( ) ; , <= >=
```

What you need to do

- build a scanner that will take an input file (LITTLE source program) and output a list of all the tokens in the program.
  - For each token, you should output the token type (e.g., OPERATOR) and its value (e.g., +).
- Inputs/outputs (i.e. testcases) are provided.
  - Your outputs need to match our outputs *exactly* (they will be automatically compared using *diff*, though whitespace will be ignored).

OCF demo

- http://oxiago.com/compiler/msu/
- Log in using: NetID
  - Check whether your group members are accurate.
- Off-campus: use a VPN