Tabulate
Final Presentation

Source lines of code:

```
src:
---
   68 CMakeLists.txt
  176 driver.cc
  317 lex.l
   32 main.cc
 1013 parser.y
    5 translation.cc
   12 types.cc
 1623 total
```

```
runtime:
-------
   30 CMakeLists.txt
  437 any.cc
  127 helper.cc
  646 inbuilt.cc
   80 include/any.hh
   27 include/helper.hh
   87 include/inbuilt.hh
   30 include/runtime.hh
   52 include/state.hh
  124 include/types.hh
   11 state.cc
   19 test.tblt
  122 translated.cc
  256 types.cc
 2048 total
```

```
include:
-------
  108 driver.hh
    0 runtime_env.hh.in
  163 symtab.hh
    6 tabulate.hh
   12 translation.hh
   71 types.hh
  360 total

4031 total
```

Purpose

- Spreadsheets are an integral part of our lives. Whether it comes to creating timetables, bookkeeping possessions or tabulating marks, it is difficult to imagine life without spreadsheets
- With high level programming constructs to abstract the implementation of seemingly complex operations, Tabulate makes it possible to program your spreadsheet
- Unlike most popular spreadsheet softwares like Microsoft Excel and Google Sheets which are What You See Is What You Get (WYSIWYG) editors, Tabulate offers a very good programming interface.

Contribution

- Gautam Singh:
  - Semantic checks
  - Implementation of inbuilt data types and inbuilt functions
  - Developing build system and testing
  - Ideation
- Anshul Sangrame:
  - Implemented runtime type checks for dynamic typing
  - Lexer
  - Translation and code generation part
  - Integrating Lexer and parser
- Varun Gupta:
  - Parser
  - Semantic Analysis
  - Language Spec
  - Implementation of inbuilt functions
- Ideas -> Implementation -> Integration
- GitHub Repository Link: https://github.com/goats-9/cs3423-project

Implementation details

- **Languages Used**
  - **Compiler**: C++
  - **Target language**: C++
- **Tools Used**
  - **Build System**: CMake
  - **Lexer**: Flex (C++)
  - **Parser**: Bison (C++)
  - **Code Generator**: C++
- **Number of test cases**: 24

INPUT

```
fun main() {

      let arr = [[1,2,”hello
world”],[3,4,5],[“test1”,”test2”,”test3”]];

      let res  = ADD(arr[0][0],arr[1][2]);

      DISP(res); let tb = new table();

      tb.assign((0:2~1,0:2~1),arr);

      tb.write(“out.csv”, “,”);

}
```

OUTPUT

stdout

```
6
```

out.csv

```
1,2,hello world

3,4,5

test1,test2,test3
```