Monash University FIT2014 Assignment 2

Due: 11:55pm, Friday 4 October 2024

1. Introduction

In this assignment, you will:

- Implement a lexical analyser using lex (Problem 3).
- Implement parsers using lex and yacc (Problems 1 - 6).
- Program a Turing mach
- Learn about some asp (Problems 2 - 6).
- Practise skills related t ges (Problem 8).

2. Assignment

- Start working early and ssible before week 10.
- Don't be deterred by th al and documentation.

3. How to Begi

1. Download the workber
2. Create a new Ed Work
3. Edit the `student - id` file with your name and student ID.
4. Open a terminal and change into the `asgn2` directory.

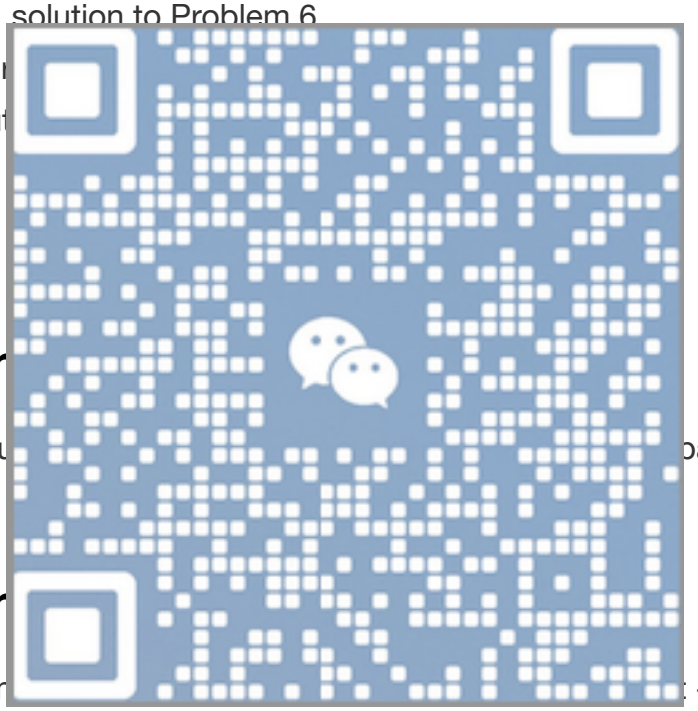
4. Files in the Directory

- `plus - times - power.l` : Starting point for some lex files.
- `plus - times - power.y` : Starting point for some yacc files.
- `quant.h` : Should remain unaltered.
- `prob6.awk` : An awk program for a specific task.

5. Submission Requirements

Your submission must include:

- `student – id` file edited with your name and student ID.
- `prob1.l` : Obtained by modifying a copy of `plus – times – power.l` .
- `prob2.pdf` : Your solution to Problem 2 (Context - Free Grammar for QCIRC).
- `prob3.l` : Obtained by modifying a copy of `plus – times – power.l` .
- `prob4.l` : Obtained by modifying a copy of `prob3.l` .
- `prob4.y` : Obtained by modifying a copy of `plus – times – power.y` .
- `prob5.l` : Obtained by modifying a copy of `prob4.l` .
- `prob5.y` : Obtained by modifying a copy of `prob4.y` .
- `prob6.txt` : Ten lines, solution to Problem 6
- `prob7.tm` : Tuatara Tu
- `prob8.pdf` : Your solu



6. Assignment

Problem 1 (2 mar

Construct `prob1.l` to be u parser for PLUS - TIMES - POWER.

Problem 2 (3 mar

Write a Context - Free Gra {I, H, X, Y, Z, CNOT, TOF, *, ⊗, (,) } and save it as `prob2.pdf` .

Problem 3 (3 marks)

Construct a lex file `prob3.l` starting from `plus – times – power.l` to build a lexical analyser for QCIRC.

Problem 4 (6 marks)

- Make a copy of `prob3.l` called `prob4.l` and modify it for use with yacc.
- Construct a yacc file `prob4.y` from `plus – times – power.y` .
- Build a parser for QCIRC that can evaluate expressions.

Problem 5 (5 marks)

- Make copies of `prob4.l` and `prob4.y` called `prob5.l` and `prob5.y` respectively.
- Modify them further to build a parser for QUANT.

Problem 6 (5 marks)

- Use the `prob6.awk` program to convert your student ID to a quantum register expression.
- Copy the expression into `prob6.txt` as the first line.
- Run your parser on the expression and append the result to `prob6.txt` .

Problem 7 (7 marks)

Build a Turing machine in T[illegible]nction and save it as `prob7.tm` .

Problem 8 (9 mar

- Prove there is no Unive[illegible]ma for Regular Languages.
- Prove there is no Unive[illegible]g lemma for Context - Free Languages. Save the p[illegible]

7. Lex and Yac

Lex

- An input file to lex ends in `.l` and has three parts: definitions, rules, C code.
- The `plus_times_power.l` file is used as an example. It identifies tokens like nonnegative integers (NUMBER), `Power` (POWER), and specific characters.
- When lex runs on the file, it generates `lex.yy.c` which can be compiled to create a lexical analyser.

Yacc

- An input file for yacc ends in `.y` and has three parts: declarations, rules, programs.
- The `plus_times_power.y` file is used as an example. It has a grammar for PLUS - TIMES - POWER and declarations for tokens and nonterminals.

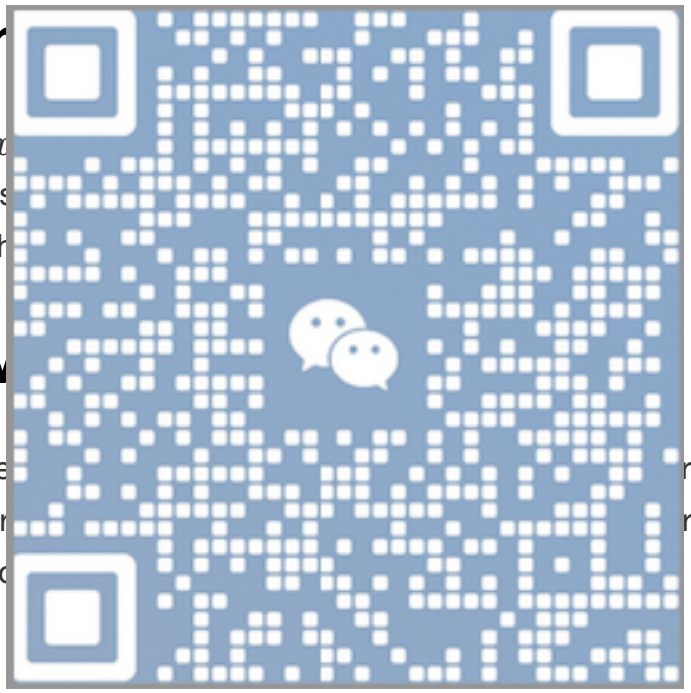
- To generate a parser, you need to modify `prob1.l` and use `plus - times - power.y` along with some compilation steps.

8. Quantum Circuits and Registers

- A quantum computer uses quantum physics for computation.
- It has a register that stores information in a superposition and a quantum circuit expression that transforms the register.
- The languages QCIRC, QREG, and QUANT are defined to describe quantum expressions.
- QCIRC is for valid quantum circuit expressions, QREG for valid quantum register expressions, and QUANT is their union.

9. The Number

- Input: A sequence k, x
- Output: x_k encoded as
- A Turing machine for th



10. Universal M

- Universal regular expre re defined.
- The goal is to find expr relevant strings for a given set of regular expressio