CS3423 Lab Exam

Aug 22 Semester

November 19th, 2022

Time: 4 hours Total: 100 marks

Instructions

- You are allowed to access flex/bison manuals ONLY through man pages that are available locally (use **info (man) flex/bison** command).
- The Internet is allowed ONLY to submit your work at the end of the exam. You should inform TAs while submission. Usage of the internet during the course of the exam is **strictly** prohibited. It is assumed that you have read the "**Rules and Logistics**" Document and are familiar with the Plagiarism policy of CSE@IITH.
- Any electronic device apart from laptops that you use for taking this exam is prohibited.
- Any methods to access solutions through unfair means would result in immediate granting of an **FR grade** in the entire course.
- You can specify any additional assumptions your implementation makes and/or status (if incomplete) in the **README** file bundled along with the solution.
    - Do note that if you are submitting only a partial solution, your chances of getting partial marks increases if you document your code and give a good README.
- Place all your code files and your readme file in the provided directory structure. *Use submit.sh (bash submit.sh <roll_no>) script to generate a compressed file that you can upload in the Google Classroom.* (Eg: bash submit.sh CS20BTECH00000).
- You can assume that the input is error-free. Meaning, there is no need to do error handling.

Q1 Mini-Doxygen: Mini Doxygen generator for C++ programs

(Difficulty: 4*; 55 marks)

Doxygen is a free software tool that can generate documentation from annotated source codes. As per the developers from Wikipedia:

> *Doxygen is a famous documentation generator and static analysis tool for software source trees. When used as a documentation generator, Doxygen extracts information from specially-formatted comments within the code. When used for analysis, Doxygen uses its parse tree to generate diagrams and charts of the code structure. Doxygen can cross reference documentation and code, so that the reader of a document can easily refer to the actual code.*

You would have come across Doxygen-generated webpages/documentation in huge projects like LLVM.

An example Doxygen page corresponding to the LoopUnrolling class of LLVM is shared with you in the shared directory. You can open the html file in the browser to understand how Doxgen works.

In this question, you will be developing a ***mini Doxygen generator*** (**Mini-Doxygen**) that will take a single C++ header file (`filename.h`) as input, and generate the corresponding documentation in a markdown file (`filename.md`).

Assumptions:

- Input - A minimal form of C++ header file
  - The header file would contain only declarations and no definitions.
  - Specifically, the header will have a set of includes followed by class declarations in that order. (Of course, with comments!)
  - Declaration of any child class would follow only after the deceleration of its parent class
  - All variables and function declarations will be within the class body.
- Types
  - Support only scalar integer and void types.
- Access modifiers (public, private, and protected)
  - Can occur in any order but at most once: i.e., all the members/attributes of a particular access modifier in the given program would occur together.

    - Access modifiers of Constructors need not be tracked: i.e., access modifier information of the constructor will not be used while generating documentation
- Comments - Two types of comments are supported
    - Doxygen comments: these comments will be used by Doxygen to generate documentation. These will be single-line comments starting with three backslashes followed by a whitespace.
        - For example: `/// This is a Doxygen comment`
    - C++ Single-line comments: these will be regular single-line comments supported by C++. These comments will be ignored by Doxygen.
- Public, private, and protected inheritance of classes are supported.
    - A child class can inherit at most one parent class per access modifier type
        - For example: a class can inherit one public and one protected parent class but it can not inherit from two public or two protected parent classes
- Constructors and functions can take any number of arguments. But, the number of arguments is a non-variable number; it is compile-time determinable.
- No need to support these special constructs of C++:
    - Virtual, friends, pointers, lambda functions, namespaces, …

Input:

- The Input will have the following parts:
    - Includes: one or more headers included in the input header file
    - Class declarations: one or more class declarations
    - Doxygen comments: one or more Doxygen comments above the class declarations
    - Constructor: one or more constructor declarations inside the class body
    - Variables: one or more variable declarations under their respective access modifier labels
    - Functions: one or more function declarations under their respective access modifier labels

Output:

- **Part 1 [15 marks]:** A dot file representing the inheritance diagram of the classes in the input
    - **Dot file format:** *Example files are given for your reference*
        - Do not change any other field except the nodes and edges as shown in the examples
        - Use red, blue, and green color edges to show private, protected, and public inheritance of classes respectively
- **Part 2 [40 marks]:** A markdown file with the following parts:

- Inheritance diagram in png format (generated from the .dot file using command dot -Tpng <*input-file-name*>.dot -o <*output-file-name*>.png)
- List of header files included
- Class references containing the following details for each class:
  - Doxygen comments from the input file
  - Constructors of the class
  - Private, protected, and public attributes of the class
  - Private, protected, and public member functions of the class
- **Markdown file format:** *Example files are given for your reference*
  - Use H3/heading3 (3 hashes and a whitespace) for Inheritance Diagram, Includes, and individual class reference headings.
  - All headings within each class reference should be in bold.
    - `**test**` in markdown is rendered as **test**
  - There should be one line gap between each section and subsections

Note:

1. Examples for inputs and their corresponding outputs are shared with you in the shared directory.
2. One pass of lexer and parser is sufficient to solve both parts i.e. submit only lex and yacc file.

Q2 Bhai-lang: A Translator from bhai-lang to python

(Difficulty: 2.5*; 30 marks)

bhai-lang is a toy programming language dedicated to all of our friends whom we call bhai😄.
In this question, you will develop a translator from bhai-lang to python3. Following are components supported by *bhai-lang*:

Examples	Description
`hi bhai`	**Program start:** *hi bhai* is the entry point for a program, anything above it will be ignored
`bye bhai`	**Program end:** *bye bhai* marks the end of a program, anything below it will be ignored
`bhai ye hai a = 10;` `bhai ye hai b = "two";` `bhai ye hai d = 'ok';` `bhai ye hai c = sahi;` `bhai ye hai d = galat;` `bhai ye hai e = nalla;`	**Variables and it's Type:** variables are declared using *bhai ye hai*. numbers and strings are similar to python equivalents. A null value is denoted using *nalla*. *sahi* and *galat* denote boolean values true and false respectively
`bol bhai "Hello World";` `bol bhai nalla, sahi, galat, 10, 'ok';`	**Build-in:** *bol bhai* is a used to print anything to console
`agar bhai (a < 20) {` `  bol bhai "a is less than 20";` `} nahi to bhai ( a < 25 ) {` `  bol bhai "a is less than 25";` `} warna bhai {` `  bol bhai "a is greater than or equal to 25";` `}`	**Conditional statements:** *bhai-lang* support if-else ladder construct, *agar bhai* block will execute if-condition is *sahi* , *nahi to bhai* block will execute else-if condition is *sahi* otherwise *warna bhai* block will execute if all the condition above are *galat*
`jab tak bhai (a < 10) {` `  a += 1;`	**Loop statements:** Statements inside *jab tak bhai* blocks are executed as long as a specified condition evaluates to *sahi*. If

`  agar bhai (a == 5) {` `    bol bhai "andar se bol bhai ", a;` `    agla dekh bhai;` `  }` `  agar bhai (a == 6) {` `    bas kar bhai;` `  }` `}`	the condition becomes *galat*, the statement within the loop stops executing, and control passes to the statement following the loop. Use *bas kar bhai* to break the loop and *agla dekh bhai* to continue within the loop

Assumptions:

- *bhai-lang* is case-sensitive language
- Regular rules for identifier in C applies for bhai-lang
- Regular rules for operators in C applies for bhai-lang

Input: Input will be a file containing a valid *bhai-lang* program

Output: Output should be a translation of *bhai-lang* code into python3 code

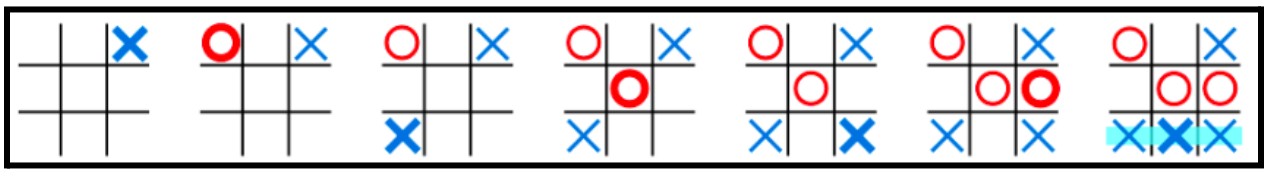
Note: Examples for Inputs and their corresponding outputs are shared with you in the shared directory.

Q3: Let's Play Tic-tac-toe

(Difficulty: 2*; 20 marks)

Tic-tac-toe: Noughts and crosses, or Xs and Os is a paper-and-pencil game for two players who take turns marking the spaces in a NxN grid with X or O. The player who succeeds in placing N of their marks in an entire row, column, or diagonal is the winner. Both players alternately place the marks X and O in one of the NxN spaces in the grid.

In the following example, for a 3x3 board, the first player (*X*) wins the game in seven steps



Configuration:

- There are two players: say A and B.
- An NxN board is in *some state*, with some filled cells with either 'X' or 'O'.
- Assume that the current state of the game is the 2nd last state: i.e., next player's move will decide the winner, or the match will be draw.
- Assume that the players will make an **optimal** move. (Each player plays to win. Not to draw.)

Problem statement:

- Given: (i) the board state(in 2nd last state), (ii) the next player (either A or B), and (iii) the next move of the next player (either 'X' or 'O').
- To do: Predict whether the player will win or the match is a draw.

Input:

- First line will contain the value of N.
- Next N lines will depict the NxN board state (blank cells of the board will be represented by **'#'**).
- Last line will contain the next player and his move.
    - Format: *<player-name><whitespace><move>*

Output:

- If the player wins, print the position of the cell where player made the move.
    - Format: *<row-number><whitespace><column-number>*
- else print the string "Draw".

Example 1:

Input:
3
X##
OO#
X##
A O

Output:
2 3

Example 2:

Input:
4
XOOX
OOXO
XXOX
O#XX
B X

Output:
Draw

ALL THE BEST
IITH-Compilers-Admin team
