From: Vandana Agarwal . (via Nalanda)
Sent: 04 February 2023 09:49
To: RAMAKANT PANDURANG TALANKAR .
Subject: 11551001318_CS F363_JAN_2023: Compiler project: Team formation, important dates and ground work

11551001318_CS F363_JAN_2023 » Forums » Announcements » Compiler project: Team formation, important dates and ground work

Compiler project: Team formation, important dates and ground work
by Vandana Agarwal . - Saturday, 4 February 2023, 9:18 AM

Team Formation: Form teams of 3-5 students on your own and register team members' names on February 6, 2023, Monday, from 4:30 p.m. to 7:30 p.m. A link will be made available on Nalanda for registering the team members names and IDs. Presence of all members is advised at the time of registration. Registering together as a team, the team members display confidence on each other and are expected to work on the compiler project together. The team members will not be changed in any circumstances throughout the semester till completion of the compiler project.

Important Dates (Stage 1)

Date of posting language specifications: **February 2, 2023**

Team formation: **February 6, 2023** [During 4:30 p.m. to 7:30 p.m]

Paper submissions

DFA neatly drawn on A3 sheet: **February 12, 2023** [During 6:30 p.m. to 7:30 p.m., Venue 6121-Z]

Complete and Modified grammar on A4 sheet: **February 19, 2023** [During 6:30 p.m. to 7:30 p.m., Venue 6121-Z]

FIRST & FOLLOW sets on A4 sheet: **February 19, 2023** [During 6:30 p.m. to 7:30 p.m., Venue 6121-Z]

Submissions through Nalanda

Coding Details in given proforma: **March 2, 2023** [During 6:30 p.m. to 8:00 p.m.]

Code Submission Due on : **March 2, 2023** [During 6:30 p.m. to 8:00 p.m.]

Ground Work

1. Read the language specifications document carefully.
2. Understand patterns and tokens.

3. Construct DFA based transition diagram (on paper) for recognizing the patterns in the source code.
4. Understand the features of the language and the grammar.
5. Complete the set of production rules to incorporate all constructs of the given language.
6. Next two weeks lectures will be on parsing techniques. Students are advised to attend classes regularly and discuss their doubts.
7. As we will progress with parsing details in the class, students can keep on preparing themselves to do the following groundwork.
8. Remove ambiguity, left recursion and perform left factoring for the productions that need to be modified.
9. Make the grammar LL(1) compatible.
10. Compute FIRST and FOLLOW sets. Verify any conflict due to ambiguity persisting in the grammar.
11. Design suitable and efficient data structure for representing and processing the language.
12. Implement lexical analyzer first and test the tokens generated.
13. Use a temporary driver to test lexical analyzer.
14. Design representation of the grammar appropriately. [To Avoid hard coding of rules, keep the grammar in one file and populate the data structure representing the grammar.]
15. Once your team members put efforts in designing the grammar rules, and submit after best efforts, you will be given support of modified grammar to correct your own grammar where ever needed.
16. Do not start coding randomly. First design the functionalities of your compiler front end and verify correctness of your DFA and grammar, then implement gradually your code.
17. Once you pour in some creativity in the design of your functionalities, you will be given support of function prototypes.
18. There will be a flexibility in the usage of these prototypes which can be changed based on your design.
19. The Linux/ Ubuntu based GCC version will be specified for the C programming language you will use for your code within a few days. Do not use just any C compiler on your own. Eventually the versions of Linux/ Ubuntu and GCC will have to be the same as those used in IPC labs.
20. Revise your concepts regularly by reading your text book chapters.
21. Feel free to fix up a time with me for a meeting whenever there is any doubt regarding design and implementation of the compiler.
22. As senior undergraduate Computer Science students, you are expected to be adaptive for making changes in your code anytime as you advance with newer concepts in compiler construction. Various data structures will be required to be redefined or modified as we progress.
23. I will conduct few special sessions on implementation issues in which we will discuss complexity of data structure design and algorithms.

See this post in context