|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SEMESTER: 5** | | **Fall-2023** | **CLASS AND SECTION** | | **BSCS-5A** |
| **TITLE OF PROJECT** | | | | | |
| **Text to Lexify Transformer** | | | | | |
| **Group Members** | | | | | |
| **S#** | **Student(s) Name** | | | **Enrollment Number** | |
| **1** | Tayyaba Imam | | | 02-134202-056 | |
| **2** | Mania Imam | | | 02-134212-013 | |
| **3** | Shavana Yousuf | | | 02-134212-022 | |
| **EXECUTIVE SUMMARY OF PROJECT PROPOSAL** | | | | | |
| The Lexify Language Project introduces a sophisticated compiler aimed at processing code written in the Lexify language—a simplified imperative programming language. This language is deliberately crafted as a subset of C++, drawing inspiration from it and incorporating modifications to enhance clarity and simplicity. In the development process, reference languages such as C++ and Python were consulted, influencing the design choices made to ensure a user-friendly and accessible experience. | | | | | |
| **PROJECT PURPOSE, SCOPE AND OBJECTIVES** | | | | | |
| **PURPOSE:**  This project is initiated with the primary purpose of empowering individuals with limited programming experience, particularly beginners, by providing a simplified and accessible entry point to coding. The project recognizes the growing importance of programming literacy and aims to make coding education inclusive and engaging. Through the creation of Lexify, a simplified imperative language inspired by a subset of C++ with modifications for clarity and simplicity, the project seeks to simplify the learning experience and facilitate a smooth transition for non-programmers and beginners into the world of coding.  **SCOPE:**  The scope of the project is well-defined and encompasses several key aspects. Firstly, the focus lies on the design and implementation of Lexify, which introduces unique data types and syntax constructs. The language is intentionally structured to enable programmers to easily modularize their code, promoting good coding practices from the outset. Additionally, the project has a specific educational focus, targeting beginners and students entering the coding field. By providing a foundational tool for learning programming, the project aims to contribute to accessible coding education and bridge the gap between non-programmers and coding enthusiasts.  **OBJECTIVES:**  The objectives of the project align with its educational purpose and defined scope. The project aims to create a user-friendly environment that prioritizes the development of coding skills over intricate syntax details. This objective is underpinned by the belief that simplicity in learning is essential for beginners. Furthermore, the project strives to enable programmers to easily modularize their code, fostering a clear and organized approach to programming. The focus on modularity and clarity is in line with the project's broader goal of providing an accessible and engaging learning experience for individuals embarking on their coding journey. | | | | | |
| **PROJECT DESCRIPTION (most important part)** | | | | | |
| The language we are compiling is named "Lexify." It serves as a straightforward imperative programming language, essentially a subset of the C++ language. Drawing inspiration from C++ and Python as our reference languages, we have implemented modifications to enhance its clarity, broaden its scope, and simplify its structure. It's important to note that our language is not intended for realistic production use; instead, it is designed to enable programmers to break down entire programs into smaller, more manageable modules and functions. Notably, our language adheres to case sensitivity, necessitating precise typing of keywords, variables, function names, and other identifiers as per our defined conventions.  Similar to other compilers, ours employs multiple layers or steps to convert a text file comprehensively into "Lexify" code. These layers are instrumental in ensuring the accurate transformation of the text file, maintaining a systematic and structured approach throughout the conversion process. The error is checked on each part or layer. The sections include:   1. Lexical Analyzer 2. Parser or Syntax Analyzer 3. Scanner 4. Intermediate Code generator 5. Code Optimization 6. Code Generation   **Lexical Analyzer:** It will take each line of code and spit it into chunks or basic units of data. All the chunks or piece of information has some meaning or definition to it. It is labeled with the identifier or definition, creating tokens. Each token is stored in a table called ‘Symbol Table’. Then all the tokens are passed on to the parser to check the syntax. It uses regular expressions to identify the basic units and can be tokenized.  **Parser:** The parser will now take the tokens, and check the CFG (Context Free Grammar) to check if the syntax is correct. It creates a parse tree. It basically checks if the pieces of the puzzle are put in right place. Once the syntax is checked, the code is handed over to the scanner.  **Scanner:** Now the context free grammar is checked for the datatypes. The scanner is a subroutine which is frequently called by an application program like a compiler. The primary function of a scanner is to combine characters from the input stream into recognizable units called tokens.  There is a symbol table which stores all the relevant information to get the desired output. Each section or part has its own errors. If the errors are not detected properly in the beginning stage, it may affect the output. So, all the possible errors and constraints are dealt at that stage.  **Tasks of Compiler:**  The main tasks performed by the Compiler are:   * Breaks up the up the source program into pieces and impose grammatical structure on them * Allows you to construct the desired target program from the intermediate representation and also create the symbol table * Compiles source code and detects errors in it. * Manage storage of all variables and codes. * Support for separate compilation * Read, analyze the entire program, and translate to semantically equivalent * Translating the source code into object code depending upon the type of machine. | | | | | |
| **TEAM PROFILE** | | | | | |
| * Tayyaba will be coding for the first layer of compiler i.e., Lexical Analyzer, token sets and type checking with documentation. * Mania will be coding for second layer of compiler i.e., Syntax Analyzer including building of CFG and parse tree with documentation. * Shavana will be Coding for third layer of compiler i.e., Semantic Analyzer, symbol table and along with documentation. | | | | | |

|  |
| --- |
| **ASSUMPTIONS AND CONSTRAINTS** |
|  |
| **Assumptions:**  Accessible Environment:  Assumption: Users have access to a suitable programming environment.  Continuous Improvement:  Assumption: The Lexify language and compiler will undergo continuous improvement.  Text Input Format:  Assumption: Input text follows the specified Lexify language format.  **Constraints:**  Time Constraints:  Constraint: Limited time availability due to concurrent academic commitments such as exams, assignments, and quizzes.  Impact: Project development may face delays or time-sensitive features may be prioritized.  Resource Allocation:  Constraint: Limited coding capacity given the need to balance project work with academic responsibilities.  Impact: Coding progress may be affected, emphasizing the necessity for efficient time management.  **SOFTWARE USED:**  Visual Studio Code , PyCharm Community Edition  **HARDWARE RECOMMENDATION**:  10 MB Free Space  4 GB RAM  **Language specification**:  Python used for programming. |
| **PROJECT DELIVERABLES NOT CHANGEABLE** |
| **Deliverables include:**   * Software Project Proposal. * Project progress * Project report * Team member’s work, as per their contribution, you should have to be honest with your future. |
| **Time Line** |
| |  |  |  | | --- | --- | --- | | **ITEM** | **DATE** | **SIGNATURE** | | Project Proposal | 06-12-2023 |  | | Project Submission |  |  | |

|  |
| --- |
| **For Teacher Use Only** |

|  |
| --- |
| **REMARKS** |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Course Teacher:** |  | **Signature** |  | **Date:** |  |
| **Lab Teacher:** |  | **Signature:** |  | **Date:** |  |