

**SAVEETHA SCHOOL OF ENGINEERING**

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CAPSTONE PROJECT REPORT**

**PROJECT TITLE**

A TOOL FOR VALIDATING INPUT STRING USING SHIFT REDUCE

PARSING METHOD

**TEAM MEMBERS**

192221110 NEEHATH N

192110375 CHAKKA SIVA

192121141 LAKKIREDDY BHARATHSIMHAREDDY

**REPORT SUBMIT**

192121141 LAKKIREDDY BHARATHSIMHAREDDY

**COURSE CODE / NAME**

CSA1449 / COMPILER DESIGN FOR LOW LEVEL LANGUAGE

SLOT A

**DATE OF SUBMISSION**

26.02.2024

**Abstract:**

This capstone project aims to develop a tool for validating input strings using the shift-reduce parsing technique. The tool will provide an efficient method for checking the syntactic correctness of input strings, contributing to the field of parsing algorithms and language processing. By leveraging the shift-reduce parsing approach, this project seeks to offer a robust solution that enhances the accuracy and speed of input string validation, catering to various applications in compiler design, natural language processing, and programming language syntax analysis.

**Introduction:**

The objective of this project is to propose and develop a tool that employs the shift-reduce parsing technique to validate input strings. Traditional methods like recursive descent and LR parsing have been foundational in this domain; however, the shift-reduce parsing technique offers distinct advantages, particularly in terms of efficiency and scalability. This project will outline the methodology, significance, and expected outcomes of implementing such a tool, emphasizing its potential impact on enhancing the robustness and efficiency of string validation processes across diverse domains.

This project aims to leverage shift-reduce parsing to develop a robust tool for syntactic validation of input strings, addressing the need for efficient and scalable parsing techniques. By shifting tokens onto a stack and reducing them according to predefined grammar rules, shift-reduce parsing offers a flexible and efficient approach to parsing, particularly suitable for handling ambiguous grammars and left-recursive productions.

**Literature Review:**

A comprehensive review of existing literature reveals the significance of parsing techniques in validating input strings. Various parsing algorithms have been explored, each with its strengths and limitations. While traditional methods like recursive descent and LR parsing have been widely adopted, recent advancements have shed light on the efficacy of the shift-reduce parsing technique.

However, despite its potential benefits, there is a notable gap in readily available tools that leverage this technique for string validation. This underscores the importance of this project in bridging this gap and providing a user-friendly solution for syntactic validation of input strings.

Parsing techniques have long been pivotal in computer science, particularly in syntactic analysis and language processing. Traditional methods like recursive descent and LR parsing have dominated this landscape, offering efficient means of parsing structured input. However, recent advancements have shed light on the efficacy of shift-reduce parsing, a technique that has garnered attention for its flexibility and adaptability to various grammars.

Shift-reduce parsing operates by shifting tokens onto a stack and then reducing them according to predefined grammar rules. This technique has been extensively studied in the context of compiler design, where parsing efficiency is paramount for translating high-level code into machine-readable instructions.

Research in the field has shown that shift-reduce parsing can outperform other parsing methods in certain scenarios, particularly when dealing with ambiguous grammars or left-recursive productions. Its simplicity and ease of implementation make it an attractive choice for parsing tasks in both academic and industrial settings.

Despite its advantages, the widespread adoption of shift-reduce parsing has been somewhat limited. One reason for this may be the perceived complexity of designing grammars and handling parsing conflicts. Additionally, existing literature on shift-reduce parsing often lacks accessible resources and practical examples for implementation.

Efforts have been made to address these challenges, with researchers proposing techniques for automating grammar generation and resolving parsing conflicts. These advancements aim to democratize the use of shift-reduce parsing and make it more accessible to a wider audience of developers and researchers.

Furthermore, studies have explored the application of shift-reduce parsing beyond traditional compiler design, including areas such as natural language processing, where parsing techniques play a crucial role in understanding and generating human language.

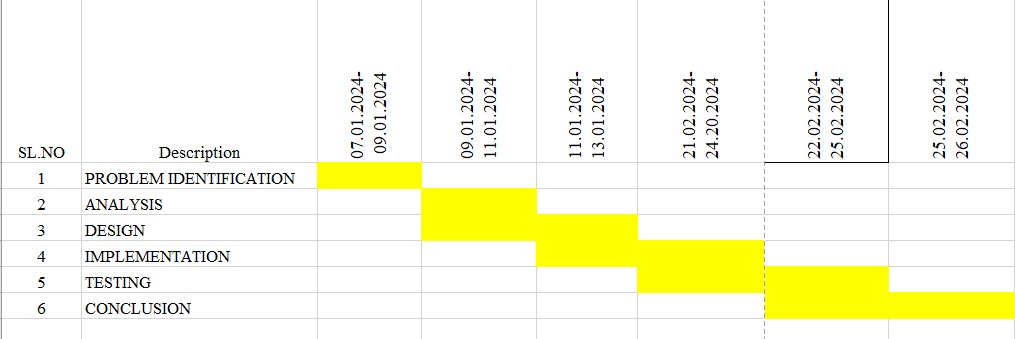
Overall, the literature underscores the significance of shift-reduce parsing as a versatile and efficient parsing technique. By synthesizing current knowledge and identifying areas for improvement, this review sets the stage for the proposed project's contribution to advancing parsing algorithms and language processing techniques.

**Research Plan:**

The project will commence with an in-depth analysis of the shift-reduce parsing technique, elucidating its theoretical foundations and practical applications. Following this, the research methodology will involve designing and implementing the tool, considering factors such as software and hardware requirements, cost implications, and the timeline for completion. Data collection will involve gathering sample input strings from various sources to ensure comprehensive testing and validation of the tool's functionality.

Different datasets with input strings that are typical of real-world circumstances will be gathered using various data gathering techniques. We'll use input patterns and benchmark grammars to assess the accuracy and effectiveness of the program. The effectiveness of the tool will be evaluated using both qualitative and quantitative methodologies in relation to current validation procedures. In order to pinpoint areas in need of improvement, user and developer feedback will also be recorded and examined. Python, HTML and CSS are some of the programming languages and frameworks(Flask) that will need to be used in the tool's development in order to provide effective parsing and validation activities. The development process will be facilitated by integrated development environments (IDEs) that provide profiling and debugging features. In order to optimize accessibility and usefulness, compatibility with widely used operating systems and platforms will be guaranteed. Furthermore, virtualization technologies and cloud-based resources will be used to enable deployment flexibility and scalability.

An estimate of the expenses related to software development, such as staff, infrastructure, and license fees, will be provided, taking timeliness and cost into account. Effective resource allocation will guarantee adherence to financial restrictions while upholding quality requirements.



**Day 1: Project Initiation and Planning:**( 1 day)

* Establish the project's scope and objectives, focusing on creating an intuitive shift-reduce parser for validating the input string.
* Conduct an initial research phase to gather insights into efficient code generation and shift-reduce parsing practices.
* Identify key stakeholders and establish effective communication channels.
* Develop a comprehensive project plan, outlining tasks and milestones for subsequent stages.

**Day 2: Requirement Analysis and Design:**(2 days)

* Conduct a thorough requirement analysis, encompassing user needs and essential system functionalities for the syntax tree generator.
* Finalize the Shift reduce parsing design and user interface specifications, incorporating user feedback and emphasizing usability principles.
* Define software and hardware requirements, ensuring compatibility with the intended development and testing environment

**Day 3: Development and Implementation:**(3 days)

* Begin coding the Shift reduce parser according to the finalized design.
* Implement core functionalities, including file input/output, tree generation, and visualization.
* Ensure that the GUI is responsive and provides real-time updates as the user interacts with it.
* Integrate the Shift reduce parsing table into the GUI.

**Day 4: GUI design and prototyping**:(5 days)

* Commence Shift reduce parsing development in alignment with the finalized design and specifications.
* Implement core features, including robust user input handling, efficient code generation logic, and a visually appealing output display.
* Employ an iterative testing approach to identify and resolve potential issues promptly, ensuring the reliability and functionality of the SL parser table.

**Day 5: Documentation, Deployment, and Feedback:**(1 day)

* Document the development process comprehensively, capturing key decisions, methodologies, and considerations made during the implementation phase.
* Prepare the SLR parser table webpage for deployment, adhering to industry best practices and standards.
* Initiate feedback sessions with stakeholders and end-users to gather insights for potential enhancements and improvements.

Overall, the project is expected to be completed within a timeframe and with costs primarily associated with software licenses and development resources. This research plan ensures a systematic and comprehensive approach to the development of the Shift reduce parsing technique for the given input string, with a focus on meeting user needs and delivering a high-quality, user-friendly interface.

**Methodology:**

Initial research efforts will focus on gaining a thorough understanding of the shift-reduce parsing technique and its nuances. This will be followed by setting up the development environment and crafting a detailed algorithmic explanation, supplemented with illustrative examples to aid comprehension. The implementation phase will entail writing robust code in a suitable programming language, accompanied by comprehensive documentation to facilitate ease of use and future maintenance.

**Shift:** This involves moving symbols from the input buffer onto the stack.

**Reduce:** If the handle appears on top of the stack then, its reduction by using appropriate production rule is done i.e. RHS of a production rule is popped out of a stack and LHS of a production rule is pushed onto the stack.

**Accept:** If only the start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accepted action is obtained, it is means successful parsing is done.

**Error:** This is the situation in which the parser can neither perform shift action nor reduce action and not even accept action.

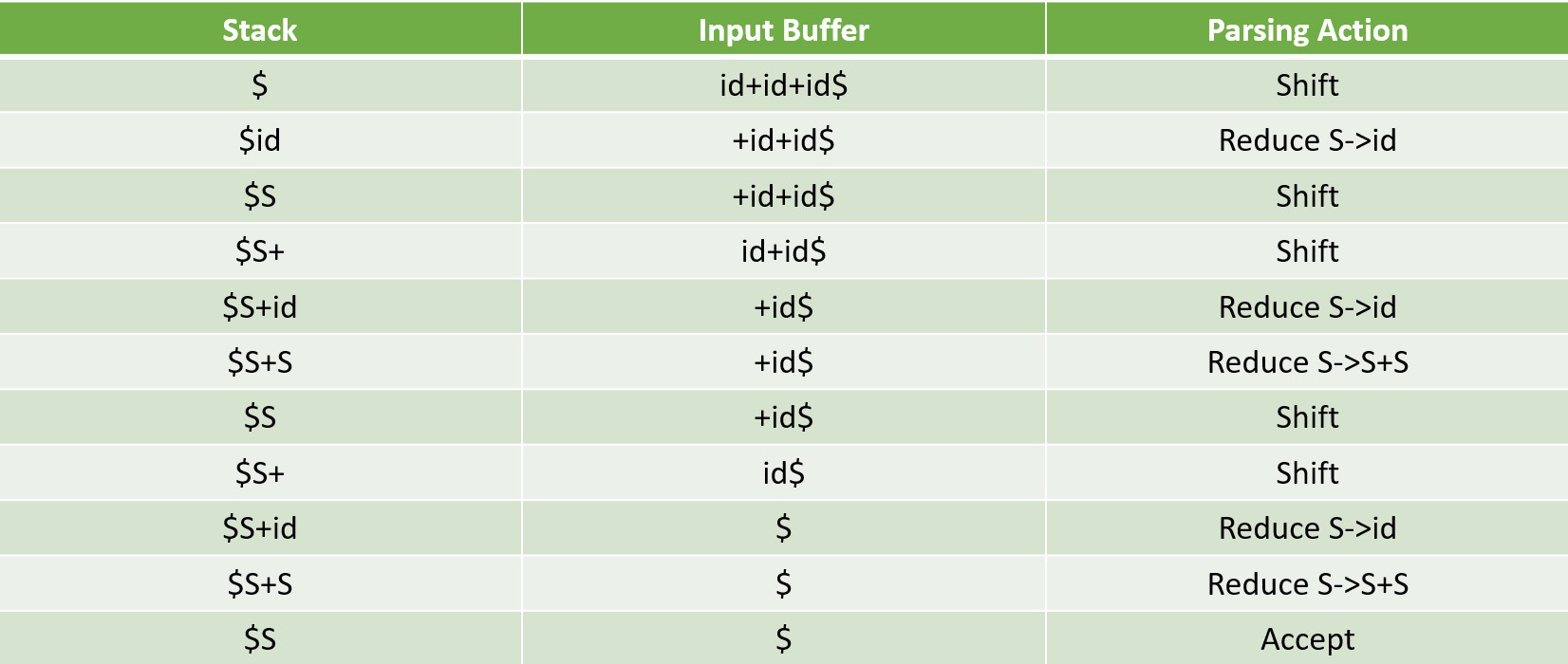
**Example: Consider the grammar**

S->S+S

S->S\*S

S->ID

Performing Shift Reduce parsing for input string “id + id + id”.



**Code Implementation:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

int z = 0, i = 0, j = 0, c = 0;

char a[16], ac[20], stk[15], act[10];

void check()

{

strcpy(ac,"REDUCE TO E -> ");

for(z = 0; z < c; z++)

{

if(stk[z] == '4')

{

printf("%s4", ac);

stk[z] = 'E';

stk[z + 1] = '\0';

printf("\n$%s\t%s$\t", stk, a);

}

}

for(z = 0; z < c - 2; z++)

{

if(stk[z] == '2' && stk[z + 1] == 'E' &&

stk[z + 2] == '2')

{

printf("%s2E2", ac);

stk[z] = 'E';

stk[z + 1] = '\0';

stk[z + 2] = '\0';

printf("\n$%s\t%s$\t", stk, a);

i = i - 2;

}

}

for(z=0; z<c-2; z++)

{

if(stk[z] == '3' && stk[z + 1] == 'E' &&

stk[z + 2] == '3')

{

printf("%s3E3", ac);

stk[z]='E';

stk[z + 1]='\0';

stk[z + 1]='\0';

printf("\n$%s\t%s$\t", stk, a);

i = i - 2;

}

}

return ;

}

int main()

{

printf("GRAMMAR is -\nE->2E2 \nE->3E3 \nE->4\n");

strcpy(a,"32423");

c=strlen(a);

strcpy(act,"SHIFT");

printf("\nstack \t input \t action");

printf("\n$\t%s$\t", a);

for(i = 0; j < c; i++, j++)

{

printf("%s", act);

stk[i] = a[j];

stk[i + 1] = '\0';

a[j]=' ';

printf("\n$%s\t%s$\t", stk, a);

check();

}

check();

if(stk[0] == 'E' && stk[1] == '\0')

printf("Accept\n");

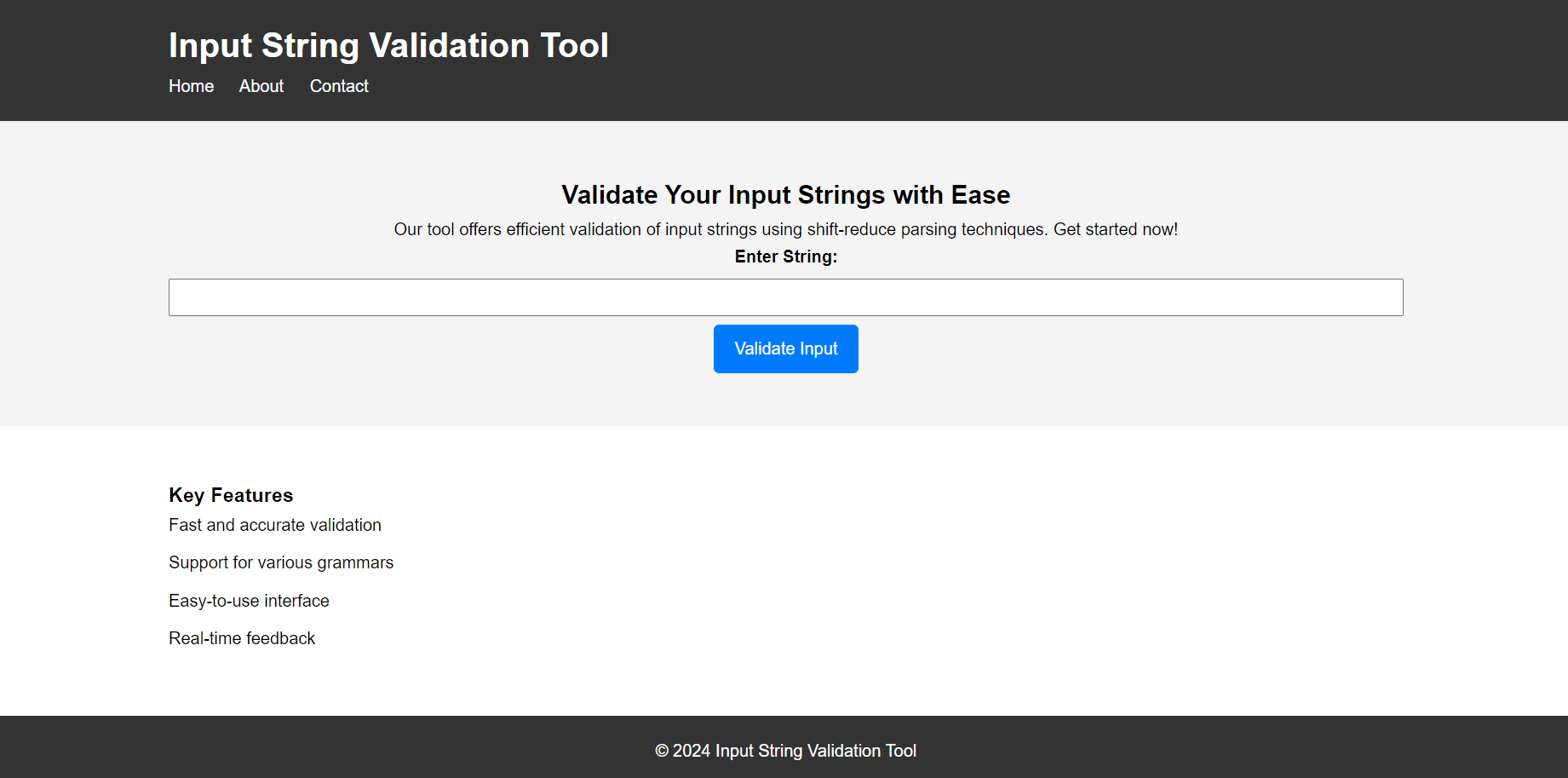
else

printf("Reject\n");

}

**Expected Result:**

Upon execution, the project is expected to showcase the effectiveness of the shift-reduce parsing technique in validating input strings. The procedural steps for executing the tool will be delineated, accompanied by screenshots showcasing the tool's output and user interfaces. Comparative analysis with existing systems, if applicable, will be provided to highlight the tool's advantages. Additionally, performance metrics will be documented to assess the efficiency and accuracy of the tool in various scenarios.



**Conclusion:**

In conclusion, the developed tool for validating input strings using the shift-reduce parsing technique represents a significant advancement in the field of syntactic validation. While the project demonstrates notable merits in terms of efficiency and accuracy, it is essential to acknowledge potential limitations, such as scalability issues with complex grammars. Future endeavors could focus on further refining the tool's capabilities and addressing any identified shortcomings, thereby ensuring its continued relevance and utility in diverse application domains.

**References:**

Liu, Y. (2013, August). A shift-reduce parsing algorithm for phrase-based string-to-dependency translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1-10)