Problem: Shift-Reduce Parsing

Aim Write a C program to implement Shift-Reduce Parser.

Theory

Shift-reduce parsing is a method for syntax analysis that constructs the parse tree on seeing an input string beginning at the leaves and working towards the root.

At each step it attempts to reduce a substring from the input by replacing it with the right side of a grammar production, thus attempting to reach the start symbol of the grammar.
At the end of the operation of the shift-reduce parser there can be traced in reverse the rightmost derivation of the input string according to the grammar.

Example: Consider the grammar

$$S \rightarrow aABe$$
$$A \rightarrow Abc \mid b$$
$$B \rightarrow d$$

And the input string: *abbcde*

This sentence can be reduced to S by the following steps:

abbcde
*a*A*bcde*
*a*A*de*
*a*AB*e*
S

This is a sequence of four reductions:

$$S \Rightarrow_{\text{rm}} aABe \Rightarrow_{\text{rm}} aAde \Rightarrow_{\text{rm}} aAbcde \Rightarrow_{\text{rm}} abbcde$$

Handles

A handle of a string is a substring that matches the right side of a production whose reduction to the nonterminal on the left represents one step along the reverse of a rightmost derivation.

A *handle* of a right-sentential form γ is a production $A \rightarrow \beta$ and a position in γ where the string β may be found and replaced by A to produce the previous sentential form in the rightmost derivation of γ.

in $S \Rightarrow^*_{rm} \alpha A w \Rightarrow \alpha\beta w$

the handle of $\alpha\beta w$ *is* $A \rightarrow \beta$ in position α

Example:

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$E \rightarrow \mathbf{id}$

Rightmost derivation:

$$\begin{aligned} E &\Rightarrow_{rm} \underline{E + E} \\ &\Rightarrow_{rm} E + \mathit{E} * \mathit{E} \\ &\Rightarrow_{rm} E + E * \underline{\mathbf{id}_3} \\ &\Rightarrow_{rm} E + \underline{\mathbf{id}_2} * \mathbf{id}_3 \\ &\Rightarrow_{rm} \underline{\mathbf{id}_1} + \mathbf{id}_2 * \mathbf{id}_3 \end{aligned}$$

Another rightmost derivation:

$$\begin{aligned} E &\Rightarrow_{rm} \underline{E * E} \\ &\Rightarrow_{rm} E * \underline{\mathbf{id}_3} \\ &\Rightarrow_{rm} \underline{E + E} * \mathbf{id}_3 \\ &\Rightarrow_{rm} E + \underline{\mathbf{id}_2} * \mathbf{id}_3 \\ &\Rightarrow_{rm} \underline{\mathbf{id}_1} + \mathbf{id}_2 * \mathbf{id}_3 \end{aligned}$$

Stack Implementation of Shift-Reduce Parsing

There are two problems to solve in parsing by handle pruning:

- how to locate the substring to be reduced in the right-sentential form;

- Which production to choose in order to implement the reduction.

A convenient way for dealing with such difficulties is to design shift-reduce parser which uses a stack to hold the grammar symbols and an input buffer to hold the string to be parsed.

The *shift-reduce parser* operates by shifting zero or more symbols onto the stack until a handle appears on the top. The parser then reduces the handle to the left side of the corresponding production. This process continues until an error occurs or the start symbol remains on the stack.

The *shift-reduce parser* performs four operations:

- *shift* - the next input symbol is shifted onto the top of the stack;
- *reduce* - the parser knows the right end of the handle is at the top of thestack. It must then locate the left end of the handle within the stack and decide with what nonterminal to replace the handle;
- *accept* - the parser announces successful completion of parsing;
- *error* - the parser discovers that a syntax error has occurred.

Examples: Successive steps in rightmost derivations:

a) $S \Rightarrow^*_{rm} \alpha A z \Rightarrow_{rm} \alpha\beta B yz \Rightarrow_{rm} \alpha\beta\gamma yz$

	Stack	Input
1)	$ $\alpha\beta\gamma$	*yz$*
2)	$ $\alpha\beta B$	*yz$*
3)	$ $\alpha\beta B y$	*z$*

b) $S \Rightarrow^*_{rm} \alpha B x A z \Rightarrow_{rm} \alpha B xyz \Rightarrow_{rm} \alpha\gamma xyz$

	Stack	Input
1)	$ $\alpha\gamma$	*xyz$*

2) $\$\ \alpha B x y$ $z\$$

Example: The shift-reduce parser for the context-free grammar

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow \textbf{id}$

performs the following steps when analyzing the input string: $\textbf{id}_1 + \textbf{id}_2 \ \textbf{*}\ \textbf{id}_3$

Stack	Input	Action
$\$$	$\textbf{id}_1 + \textbf{id}_2 \ \textbf{*}\ \textbf{id}_3\ \$$	shift
$\$\ \textbf{id}_1$	$+ \textbf{id}_2 \ \textbf{*}\ \textbf{id}_3\ \$$	reduce by $E \rightarrow \textbf{id}$
$\$\ E$	$+ \textbf{id}_2 \ \textbf{*}\ \textbf{id}_3\ \$$	shift
$\$\ E+$	$\textbf{id}_2 \ \textbf{*}\ \textbf{id}_3\ \$$	shift
$\$\ E+\textbf{id}_2$	$\textbf{*}\ \textbf{id}_3\ \$$	reduce by $E \rightarrow \textbf{id}$
$\$\ E+E$	$\textbf{*}\ \textbf{id}_3\ \$$	shift
$\$\ E+E*$	$\textbf{id}_3\ \$$	shift
$\$\ E+E*\ \textbf{id}_3$	$\$$	reduce by $E \rightarrow \textbf{id}$
$\$\ E+E*E$	$\$$	reduce by $E \rightarrow E*E$
$\$\ E+E$	$\$$	reduce by $E \rightarrow E+E$
$\$E$	$\$$	accept