

## Assignment of Compiler Design

1. Write a C program that read the following string:  
**"Md. Tareq Zaman, Part-3, 2011"**
  - a) Count number of words, letters, digits and other characters.
  - b) Separates letters, digits and others characters.
2. Write a program that read the following string:  
**"Munmun is the student of Computer Science & Engineering"**.
  - a) Count how many vowels and Consonants are there?
  - b) Find out which vowels and consonants are existed in the above string?
  - c) Divide the given string into two separate strings, where one string only contains the words started with vowel, and another contains the words started with consonant.
3. Write a program that abbreviates the following code:  
**CSE-3141 as Computer Science & Engineering, 3<sup>rd</sup> year, 1<sup>st</sup> semester, Compiler Design, Theory.**
4. Build a lexical analyzer implementing the following regular expressions:
 

---

 Integer variable = (i-nl-N)(a-zA-Z0-9)\*  
 ShortInt Number = (1-9)|(1-9)(0-9)|(1-9)(0-9)(0-9)|(1-9)(0-9)(0-9)(0-9)  
 LongInt Number = (1-9)(0-9)(0-9)(0-9)(0-9)+  
 Invalid Input or Undefined = Otherwise
 

---
5. Build a lexical analyzer implementing the following regular expressions:
 

---

 Float variable = (a-zA-Z0-9)(a-zA-Z0-9)\*  
 Float Number = 0.(0-9)(0-9)|(1-9)(0-9)\*.(0-9)(0-9)  
 Double Number = 0.(0-9)(0-9)(0-9)+|(1-9)(0-9)\*.(0-9)(0-9)(0-9)+  
 Invalid Input or Undefined = Otherwise
 

---
6. Build a lexical analyzer implementing the following regular expressions:
 

---

 Character variable = ch\_(a-zA-Z0-9)(a-zA-Z0-9)\*  
 Binary variable = bn\_(a-zA-Z0-9)(a-zA-Z0-9)\*  
 Binary Number = 0(0|1)(0|1)\*  
 Invalid Input or Undefined = Otherwise
 

---



7. Write a program to recognize C++  
 i) Keyword ii) Identifier iii) Operator iv) Constant
8. Write a program which converts a word of C++ program to its equivalent token.

**RESULT:**

Input: 646.45

Output: Float

Input: do

Output: Keyword

Input: 554

Output: Integer

Input: abc

Output: Identifier

Input: +

Output: Arithmetic Operator

9. Write a program that will check an English sentence given in **present indefinite** form to justify whether it is syntactically valid or invalid according to the following **Chomsky Normal Form**:

$S \rightarrow SUB \ PRED$

$SUB \rightarrow PN \mid P$

$PRED \rightarrow V \mid V \ N$

$PN \rightarrow Sagor \mid Selim \mid Salma \mid Nipu$

$P \rightarrow he \mid she \mid I \mid we \mid you \mid they$

$N \rightarrow book \mid cow \mid dog \mid home \mid grass \mid rice \mid mango$

$V \rightarrow read \mid eat \mid take \mid run \mid write$

10. Write a program to implement a shift reducing parsing.

11. Write a program to generate a syntax tree for the sentence  $a+b*c$  with the following grammar:

$E \rightarrow E+E \mid E-E \mid E * E \mid E / E \mid (E) \mid a \mid b \mid c$

12. Write a program which checks a validity of C++ expression derived by the following grammar:

$E \rightarrow E \ A \ E \mid (E) \mid ID$

$A \rightarrow + \mid - \mid * \mid /$

$ID \rightarrow \text{any valid identifier} \mid \text{any valid integer}$

**RESULT:**

Input: Enter a string : 2+3\*5

Output: VALID

Input: Enter a string : 2+\*3\*5

Output: INVALID



13. Write a program to generate FIRST and FOLLOW sets using a given CFG.

14. Write a program to generate a FOLLOW set and parsing table using the following LL(1) grammar and FIRST set:

Grammar	FIRST set
$E \rightarrow TE'$	{id, (}
$E' \rightarrow +TE' \mid \epsilon$	{+, $\epsilon$ }
$T \rightarrow FT'$	{id, (}
$T' \rightarrow *FT' \mid \epsilon$	{*, $\epsilon$ }
$F \rightarrow (E) \mid id$	{id, (}

15. Write a program to generate a parse tree of predictive parser using the following parsing table:

	id	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$	$E' \rightarrow \epsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$
F	$F \rightarrow id$			$F \rightarrow (E)$		

16. Write a program that converts the C++ expression to an intermediate code of Post-fix notation form.

**RESULT:**

*Input:*

Enter infix expression : ( A - B ) \* ( D/E)

*Output:*

Postfix : AB - DE / \*

17. Write a program that converts the C++ statement to an intermediate code of Post-fix notation form.

**RESULT:**

*Input:*

Enter infix statement : if a then if c-d then a+c else a\*c else a+b

*Output:*

Postfix : acd - ac + ac \* ? ab + ?