**TOTAL NUMBER OF VOWELS**

1. START
2. INITIALIZE TWO COUNTERS VOW\_COUNT=0, CONST\_COUNT=0
3. READ I/P STRING FROM USER
4. FOR EACH CHARACTER IN I/P STRING YYLEX CHECKS:

4.1 IF CHARACTER MATCHES [AEIOU,aeiou]INCREMENTVOW\_COUNT BY1

4.2 IF CHARACTER MATCHES [A-Z,a-z] INCREMENT CON\_COUNT BY 1

4.3 IGNORE ANY OTHER CHARACTERS LIKE SPACES ,DIGITS OR SYMBOL

1. WHEN END OF I/P IS REACHED YYLEX () FINISHES SCANNING
2. PRINT THE TOTAL NUMBER OF VOWELS AND TOTAL NUMBER OF CONSONENTS
3. STOP

**YACC INDENIFIER STMENT WITH A LETTER**

1. START
2. INITIALIZE VALID =1
3. INPUT A STRING AS A IDENTIFIER
4. LEX PART : LEX READS I/P CHARACTER

4.1 IF [a-zA-Z\_][a-zA-Z\_0-9]\*

4.2 RETURN LETTER

4.3 IF [0-9]

4.4 RETURN DIGIT

4.5 IF .

4.6 RETURN YYTEXT[0]

4.7 /N RETURN 0

1. PARSE THE TOKEN USING YACC ACCORDING TO THE RULES:

5.1 IDENDIFIER MUST BEGIN WITH LETTER OR UNDERSCORE

5.2 AFTER FIRST CHARACTER IT CAN HAVE LETTERS OR DIGITS

5.3 NO OTHER SYMBOL OR SPACES ARE ALLOWED

1. IF YACC FINDS THE I/P ANY ERROR IT CALLS THE YERROR() FUNCTION
2. DISPLAY THE RESULT AFTER YACC FINISHING THE PARSING AND IF VALID=1 PRINT IT IS AN IDENDIFIER ELSE PRINT NOT IDENDIFIER
3. STOP

**ε – closure of all states of any given NFA with ε transition.**

1. START
2. OPEN 1/P FILE IN READ MODE
3. READ THE NUMBER OF STATES AND STORE THEM IN ARRAY STATES
4. FOR EACH STATE IN STATES REPEAT

4.1 INITALIZE AN INDEX I=0, ADD THE STATE TO EPSILON CLOSURE

4.2 READ TRANSITIONS AS 3 STRINGS STAE1 ,I/P,STATE2

4.3 FOR EACH LINE :

4.3.1 IF STATE1 MATCH CURRENT STATE AND 1/P IS ‘E’

4.3.2 ADD STAE2 TO CLOSURE LIST

4.3.3 COPY STATE2 TO STATE

4.4 STOP WHEN EOF IS REACHED

1. PRINT THE STATES
2. RESET THE FILE POINTER
3. CLOSE FILE
4. STOP

**RECURSIVE DECENT PARSER**

**e() start parsing ,eprime() handle +operator, t() handle term ,tprime() handle \*,check() brackets**

1.START

2.INITIALIZE COUNT =0 (TRACK CURRENT POSITION),FLAG=0(FOR ERROR)

3.READ ALGEBRIC EXPRESSION AS A STRING

4.CALL E() FUNCTION - START SYMBOL OF GRAMMER

4.1 GRAMMAR USED :

E-> T E`

E`->TE` |EEPSILON

T`->FT`

T`->\*FT`|EPSILON

F-> (E) |ID

5. E() CALLS T() THEN CALLS EPRIME()

6. T() CALLS CHECK() THEN CALLS TPRIME()

7.TPRIME()

7.1 IF CURRENT CHARACTER IS \* INCREMENT COUNT

7.2 CALL CHECK()

7.3 RECURSIVELY CALL TPRIME() FOR MORE \* OPERATORS

8.CHECK()

8.1 IF CURRENT CHARACTER IS ALPHANUMERIC INCREMENT COUNT

8.2 ELSE IF CURRENT CHARACTER IS ‘(‘ RECURSIVELY CALL E() TILL ‘)’

8.3IF CLOSING BRACKET NOT FOUND SET FLAG ==1

8.4 IF NO VALID CONDITION SET FLAG==1

9.EPRIME()

9.1 IF CURRENT CHARACTER IS + INCREMENT COUNT ,CALL T() AND THEN RECURSIVELY CALL EPRIME()

10. AFTER E() FINISHES CHECK IF COUNT = STRLEN(EXPRESSION) AND FLAG ==0 THE EXPRESSION IS VALID

11. ELSE EXPRESSION IS INVALID

12.STOP

**INTERMEDIATE CODE GENERATION (LEX+YAC)**

1.START

2.INITALIZE DIFFERENT FUNCTION PUSH(),POP(),POP2(),POP3()

3.IN LEX PART :

3.1 DEFINE PATTERS FOR ID ENDIFIERS AND NUMBERS USING TOKEN ID

3.2 FOR OPERATORS RETURN CHARACTER ITSELF

3.3 IGNORE WHITE SPACE

4. YACC PART:

4.1 DEFINE GRAMMAR RULES:

S-> ID ‘=’ E

E-> E+T|E-T|T

T->T\*F|T/F|F

F-> (E)|ID|-F

5.

5.1 PUSH() WHEN ID IS ENCOUNTERED

5.2 POP3() WHEN BINARY OPERATION LIKE +,-,\*,/

5.3 POP2() FOR URINARY OPERATIONS

5.4 POP() AT THE END AFTER ASSIGNMENT

6.. DEFINE MAIN FUNCTION

6..1 INPUT AN EXPRESSION

6.2 CALL YYPARSE() TO START PASSING USING YACC

7. IF INPUT EXPRESSION FOLLOW GRAMMAR IT PRINT SEQUENCE OF STACK OPERATION ELSE ERROR MSG

8.STOP

**IMPLEMENTATION OF CALCULATOR USING LEX AND YACC**

1.START

2.READ THE I/P EXPRESION

3.LEX PHASE : SCAN EACH CHARECTER

3.1 IF I/P HAS DIGITS CONVERT THEM INTO INTEGER AND RETURN TOKEN NUMBER

3.2 IF 1/P HAS OPRATER RETURN THE SAME CHRECTER AS TOKEN

3.3 IGNORE SPACES AND TABS

3.4 IF NEWLINE RETURN 0 TO INDICATE END OF INPUT

4.YACC PHASE : RECIEVE TOKEN FROM LEX

5. APPLY GRAMMER RULES :

EXPR : EXPR ‘+’ EXPR {$$ =$1+$3;} |

EXPR’-’EXPR{$$=$1-$3;} |

EXPR’\*’ EXPR($$=$1\*$3;)|

EXPR’/’ EXPR($$=$1/$3;)|

EXPR :’(‘EXPR’)’ {$$=$2;} |

EXPR :NUMBER {$$=$1;}

6.PRINT FINAL RESULT

7.HANDLE INVALID EXPRESSIONS

8.STOP

**CONVERT SUBSTRING ABC TO abc**

1. START
2. READ INPUT TEXT CHARECTER BY CHARECTER
3. IF I/P MATCHES PATERN [a-zA-Z]+ STORE IT IN YYTEXT

3.1 FOR EVERY I FROM 0 TO YYLEN-3 CHECK IF THE SUBSTRING AT POSITION I IS ABC .IF YES REPLACE WITH CAPITAL ABC

3.2 PRINT MODIFIED STRING

1. IF 1/P CONTAIN TAB CHARECTERS [\T]+ IGNORE THEM
2. IF I/P CONTAIN ANY OTHER CHRECTERS .\* PRENT THEM USING ECHO
3. IF NEW LINE \N FOUND PRINT IT TO MAINTAIN LINE SPACING
4. REPEAT THE ABOVE STEPS UNTIL END OF I/P IS REACHED
5. END