```
Name: ALAN D ANDOOR                                          EXPT NO: 1
DATE: 11/10/22                                               ROLL NO: 6
                                                             BATCH  :S7 CSB
*************************************************************************
                    RECURSIVE DESCENT PARSER - 1
            ----------------------------

#include<stdio.h>
char str[20];
int i=0,x,y;
int A(){
    if(str[i]=='a'){
        ++i;
        if(str[i]=='b'){
            ++i;
            return 1;
        }
        return 1;
    }
    return 0;
}
int S(){
    if(str[i]=='a'){
        ++i;
        y=A();
        if(y==1 && str[i]=='d')
            return 1;
        else
            return 0;
    }
    return 0;
}

int main(){
    printf("Enter String:");
    scanf("%s",str);
    x=S();
    if(x==1)
        printf("Accepted");
    else
        printf("Not accepted");
}


*************************************************************************
OUTPUT
------

Enter String:aabd
Accepted

Enter String:abd
Not accepted
```

```
Name: ALAN D ANDOOR                                          EXPT NO: 2
DATE: 21/10/22                                               ROLL NO: 6
                                                             BATCH  :S7 CSB
*************************************************************************
                    RECURSIVE DESCENT PARSER - 2
                    ----------------------------

#include<stdio.h>
#include<string.h>
#include<ctype.h>
char input[30];
```

```
int i=0,e=0;
void E();
void Edash();
void T();
void Tdash();
void F();
int main(){
        printf("Enter string:");
        scanf("%s",input);
        E();
        if(strlen(input)==i && e==0)
                printf("Accepted\n");
        else    printf("Not Accepted\n");
        return 0;
}
void E(){
        T();
        Edash();
}
void Edash(){
        if(input[i]=='+'){
                i=i+1;
                T();
                Edash();
        }
        else    return;
}
void T(){
        F();
        Tdash();
}
void Tdash(){
        if(input[i]=='*'){
                i++;
                F();
                Tdash();
        }
        else    return;
}

void F(){
        if(input[i]=='('){
                i++;
                E();
                if(input[i]==')')
                        i++;
                else    e=1;
        }
        else if(isalnum(input[i]))
                i++;
        else    e=1;
}

*************************************************************************
OUTPUT
------
Enter string:(5*6)+(6+8)
Accepted

Enter string:5+(5-8)
Not Accepted
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 3
DATE: 29/10/22                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
*************************************************************************
```

```
                              LEXICAL ANALYZER
                              ----------------

#include<stdio.h>
#include<string.h>
#include<ctype.h>
void main(){
    char token[50],c;
    int i=0;
    FILE *fp=fopen("input.txt","r");
    c=fgetc(fp);
    while(c!=EOF){
        if(c=='/'){
            c=getc(fp);
            if(c=='/'){
                do{
                    c=getc(fp);
                  }while(c!='\n' && c!=EOF);
            }
            else{
                printf("\n/\tOPERATOR");
            }
        }
        else if(isalpha(c) && c!=' ' && c!='\n'){
            i=0;
            while(isalpha(c) && c!=' ' && c!='\n'){
                token[i++]=c;
                c=fgetc(fp);
            }
            token[i]='\0';
            if(strcmp(token,"void")==0 || strcmp(token,"int")==0
            || strcmp(token,"char")==0 ||
            strcmp(token,"float")==0)
                printf("\n%s\tKEYWORD",token);
            else
                printf("\n%s\tIDENTIFIER",token);
        }
        else if(c=='=' || c=='+' || c=='-' || c=='*'){
            printf("\n%c\tOPERATOR",c);
            c=fgetc(fp);
        }
        else if(c=='(' || c==')' || c=='{' || c=='}' || c==';'){
            printf("\n%c\tSPECIAL CHARACTER",c);
            c=fgetc(fp);
        }
        else if(isdigit(c)&& c!=' ' && c!='\n'){
            i=0;
            while(isdigit(c) && c!=' ' && c!='\n')
            {
                token[i++]=c;
                c=fgetc(fp);
            }
            token[i]='\0';
            printf("\n%s\tNUMBER",token);
        }
        else
        {
            c=fgetc(fp);
        }
    }
    fclose(fp);
}

************************************************************************

INPUT FILE(input.txt)
```

```
---------------------
void main()
{
int x=95; //Comment
print(x);
}

OUTPUT
------
void    KEYWORD
main    IDENTIFIER
(       SPECIAL CHARACTER
)       SPECIAL CHARACTER
{       SPECIAL CHARACTER
int     KEYWORD
x       IDENTIFIER
=       OPERATOR
95      NUMBER
;       SPECIAL CHARACTER
print   IDENTIFIER
(       SPECIAL CHARACTER
x       IDENTIFIER
)       SPECIAL CHARACTER
;       SPECIAL CHARACTER
}       SPECIAL CHARACTER
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 4
DATE: 18/11/22                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
*************************************************************************
                              FIRST & FOLLOW
                      --------------

#include<stdio.h>
#include<math.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
int n,m=0,i=0,j=0;
char a[10][10],f[10];
void follow(char c);
void first(char c);
void firstT(char c,int x,int y);
void firstF(char c,char z,int x,int y);
int main(){
        int i,z;
        char c,ch;
        printf("Enter the no of productions: ");
        scanf("%d",&n);
        printf("Enter the productions: ");
        for(i=0;i<n;i++)
        scanf("%s%c",a[i],&ch);
        do{
                m=0;
                printf("Enter elemants whose first and follow is to be found:");
                scanf("%c",&c);
                first(c);
                printf("First(%c)= ",c);
                for(i=0;i<m;i++)
                        printf("%c ",f[i]);
                printf("\n");
                strcpy(f,"");
                m=0;
                follow(c);
                printf("Follow(%c)= ",c);
                for(i=0;i<m;i++)
```

```
                        printf("%c ",f[i]);
                printf("\n\n");
                printf("Continue(0/1)?");
                scanf("%d%c",&z,&ch);
        }while(z==1);
        return 0;
}
void first(char c){
        int k;
        if(!isupper(c))
                f[m++]=c;
        for(k=0;k<n;k++){
                if(a[k][0]==c){
                        if(a[k][2]=='#')
                                f[m++]='#';
                        else if(islower(a[k][2]))
                                f[m++]=a[k][2];
                        else
                                firstT(a[k][2],k,3);
                }
        }
}
void firstT(char c,int x,int y){
        int k;
        if(!isupper(c))
                f[m++]=c;
        for(k=0;k<n;k++){
                if(a[k][0]==c){
                        if(a[k][2]=='#'){
                                if(a[x][y]!='\0')
                                        firstT(a[x][y],x,y+1);
                                else
                                        f[m++]='#';
                        }
                        else if(islower(a[k][2]))
                                f[m++]=a[k][2];
                        else
                                firstT(a[k][2],k,3);
                }
        }

}
void follow(char c){
        if(a[0][0]==c)
                f[m++]='$';
        for(i=0;i<n;i++) {
                for(j=2;j<strlen(a[i]);j++){
                        if(a[i][j]==c){
                                if(a[i][j+1]!='\0')
                                        firstF(a[i][j+1],a[i][0],i,j+2);
                                if(a[i][j+1]=='\0' && c!=a[i][0])
                                        follow(a[i][0]);
                        }
                }
        }
}
void firstF(char c,char z,int x,int y){
        int k;
        if(!isupper(c))
                f[m++]=c;
        for(k=0;k<n;k++){
                if(a[k][0]==c){
                        if(a[k][2]=='#'){
                                if(a[x][y]!='\0')
                                        firstF(a[x][y],z,x,y+1);
                                else
```

```
                                        follow(a[x][0]);
                        }
                        else if(islower(a[k][2]))
                                f[m++]=a[k][2];
                        else
                                firstF(a[k][2],a[k][0],k,3);
                }
        }

}

*************************************************************************

OUTPUT
------

Enter the no of productions: 8
Enter the productions: E=TA
A=+TA
A=#
T=FB
B=*FB
B=#
F=(E)
F=i
Enter elemants whose first and follow is to be found:E
First(E)= ( i
Follow(E)= $ )

Continue(0/1)?1
Enter elemants whose first and follow is to be found:A
First(A)= + #
Follow(A)= $ )

Continue(0/1)?1
Enter elemants whose first and follow is to be found:T
First(T)= ( i
Follow(T)= + $ )

Continue(0/1)?1
Enter elemants whose first and follow is to be found:B
First(B)= * #
Follow(B)= + $ )

Continue(0/1)?1
Enter elemants whose first and follow is to be found:F
First(F)= ( i
Follow(F)= * + $ )

Continue(0/1)?0
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 5
DATE: 18/11/22                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
*************************************************************************
                           CONSTANT PROPAGATION
                   --------------------

#include<stdio.h>
#include<string.h>
#include<ctype.h>
void input();
void output();
void change(int p,char *res);
void constant();
struct expr{
        char op[2],op1[5],op2[5],res[5];
        int flag;
}arr[10];
int n;
void main(){
        input();
        constant();
        output();
}
void input(){
        int i;
        printf("\n\nEnter the maximum number of expressions : ");
        scanf("%d",&n);
        printf("\nEnter the input : \n");
        for(i=0;i<n;i++){
                scanf("%s",arr[i].op);
                scanf("%s",arr[i].op1);
                scanf("%s",arr[i].op2);
                scanf("%s",arr[i].res);
                arr[i].flag=0;
        }
}
void constant(){
        int i;
        int op1,op2,res;
        char op,res1[5];
        for(i=0;i<n;i++){
                if(strcmp(arr[i].op,"=")==0){
                        op1=atoi(arr[i].op1);
                        op2=atoi(arr[i].op2);
                        op=arr[i].op[0];
                        res=op1;
                        sprintf(res1,"%d",res);
                        arr[i].flag=1;
                        change(i,res1);
                }
        }
}
void output(){
        int i=0;
        printf("\nOptimized code is : ");
        for(i=0;i<n;i++){
                if(!arr[i].flag){
                        printf("\n%s %s %s %s",arr[i].op,
                                arr[i].op1,arr[i].op2,arr[i].res);
                }
        }
}
void change(int p,char *res){
        int i;
        for(i=p+1;i<n;i++){
                if(strcmp(arr[p].res,arr[i].op1)==0)
                        strcpy(arr[i].op1,res);
                else if(strcmp(arr[p].res,arr[i].op2)==0)
                        strcpy(arr[i].op2,res);
        }
}
*************************************************************************

OUTPUT
------
Enter the maximum number of expressions : 5

Enter the input :
= 3 - a
= 4 - b
```

```
+ a b t1
+ a c t2
+ t1 t2 t3

Optimized code is :
+ 3 4 t1
+ 3 c t2
+ t1 t2 t3
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 6
DATE: 18/11/22                                              ROLL NO: 6
                                                            BATCH   :S7 CSB
*************************************************************************
                           SHIFT REDUCE PARSER
                           -------------------

#include<stdio.h>
#include<string.h>
int k=0,z=0,i=0,j=0,c=0;
char a[16],ac[20],stk[15],act[10];
void check();
int main(){
      puts("GRAMMAR is E->E+E \n E->E*E \n E->(E) \n E->id");
      puts("enter input string ");
      scanf("%s",a);
      c=strlen(a);
      strcpy(act,"SHIFT->");
      puts("stack \t input \t action");
      for(k=0,i=0; j<c; k++,i++,j++){
         if(a[j]=='i' && a[j+1]=='d'){
              stk[i]=a[j];
              stk[i+1]=a[j+1];
              stk[i+2]='\0';
              a[j]=' ';
              a[j+1]=' ';
              printf("\n$%s\t%s$\t%sid",stk,a,act);
              check();
         }
         else{
              stk[i]=a[j];
              stk[i+1]='\0';
              a[j]=' ';
              printf("\n$%s\t%s$\t%ssymbols",stk,a,act);
              check();
         }
      }
}
void check(){
     strcpy(ac,"REDUCE TO E");
     for(z=0; z<c; z++)
       if(stk[z]=='i' && stk[z+1]=='d'){
           stk[z]='E';
           stk[z+1]='\0';
           printf("\n$%s\t%s$\t%s",stk,a,ac);
           j++;
       }
     for(z=0; z<c; z++)
       if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='E'){
           stk[z]='E';
           stk[z+1]='\0';
           stk[z+2]='\0';
           printf("\n$%s\t%s$\t%s",stk,a,ac);
           i=i-2;
       }
     for(z=0; z<c; z++)
       if(stk[z]=='E' && stk[z+1]=='*' && stk[z+2]=='E'){
           stk[z]='E';
           stk[z+1]='\0';
           stk[z+1]='\0';
           printf("\n$%s\t%s$\t%s",stk,a,ac);
           i=i-2;
       }
     for(z=0; z<c; z++)
       if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')'){
           stk[z]='E';
           stk[z+1]='\0';
           stk[z+1]='\0';
           printf("\n$%s\t%s$\t%s",stk,a,ac);
           i=i-2;
       }
}
*************************************************************************

OUTPUT
------
GRAMMAR is E->E+E
 E->E*E
 E->(E)
 E->id

enter input string
id+id*id

stack    input          action
$id       +id*id$       SHIFT->id
$E        +id*id$       REDUCE TO E
$E+        id*id$       SHIFT->symbols
$E+id        *id$       SHIFT->id
$E+E         *id$       REDUCE TO E
$E           *id$       REDUCE TO E
$E*           id$       SHIFT->symbols
$E*id           $       SHIFT->id
$E*E            $       REDUCE TO E
$E              $       REDUCE TO E
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 7
DATE: 16/12/22                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
*************************************************************************
                              DFA MINIMIZATION
                      ----------------

#include <stdio.h>

struct node {
  int a, b, f, flag;
}
a[10];
int n, j, i, k, l, m, c, ch;
void replace(int x, int y) {
  for (i = 0; i < n; i++) {
    if (a[i].flag == 1) {
      if (a[i].a == y)
        a[i].a = x;
      if (a[i].b == y)
        a[i].b = x;
    }
  }
}
void minimize() {
  do {
    ch = 0;
    for (i = 0; i < n; i++) {
      if (a[i].flag == 1) {
        k = a[i].a;
        l = a[i].b;
        m = a[i].f;
        for (j = i + 1; j < n; j++) {
          if (a[j].flag == 1) {
            if (a[j].a == k && a[j].b == l && a[j].f == m) {
              a[j].flag = 0;
              replace(i, j);
              ch = 1;
            }
          }
        }
      }
    }
  } while (ch == 1);
}
void unreachable() {
  do {
    ch = 0;
    for (i = 1; i < n; i++) {
      if (a[i].flag == 1) {
        c = 0;
        for (j = 0; j < n; j++) {
          if (i != j && a[j].flag == 1) {
            if (a[j].a == i || a[j].b == i) {
              c = 1;
              break;
            }
          }
        }
        if (c == 0) {
          a[i].flag = 0;
          ch = 1;
        }
      }
    }
  } while (ch == 1);
}
int main() {
  printf("Enter the no of states:");
  scanf("%d", & n);
  printf("\n Enter the transition table for DFA\nState\ta\tb\n");
  for (i = 0; i < n; i++) {
    scanf("%d%d%d", & j, & k, & l);
    a[j].a = k;
    a[j].b = l;
    a[j].flag = 1;
    a[j].f = 0;
  }
  printf("\nEnter the no of Final states:");
  scanf("%d", & m);
  printf("Enter the final states:");
  for (i = 0; i < m; i++) {
    scanf("%d", & j);
    a[j].f = 1;
  }
  unreachable();
  minimize();
  printf("\nMinimized DFA\n");
  printf("State\ta\tb\n");
  for (i = 0; i < n; i++) {
    if (a[i].flag == 1)
      printf("%d\t%d\t%d\n", i, a[i].a, a[i].b);
  }
  return 0;
}

*************************************************************************

OUTPUT
------
Enter the no of states:8

Enter the transition table for DFA
State   a       b
0       5       1
1       2       6
2       2       0
3       2       6
4       5       7
5       6       2
6       4       6
7       2       6

Enter the no of Final states:1
Enter the final states:2

Minimized DFA
State   a       b
0       5       1
1       2       6
2       2       0
5       6       2
6       0       6
```

```
Name: ALAN D ANDOOR                                     EXPT NO: 8
DATE: 16/12/22                                          ROLL NO: 6
                                                        BATCH  :S7 CSB
*************************************************************************
                          INTERMEDIATE CODE GENERATION
                  ----------------------------

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#define SIZE 100
char stack[SIZE], stack1[SIZE];
int top = -1, k = 1;
void push(char item) {
  if (top >= SIZE - 1) {
    printf("\nStack Overflow.");
  } else {
    top = top + 1;
    stack[top] = item;
  }
}
char pop() {
  char item;
  if (top < 0) {
    printf("stack under flow: invalid infix expression");
    getchar();
    exit(1);
  } else {
    item = stack[top];
    top = top - 1;
```

```c
    return (item);
  }
}
int is_operator(char symbol) {
  if (symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-') {
    return 1;
  } else return 0;
}
int precedence(char symbol) {
  if (symbol == '^') {
    return (3);
  } else if (symbol == '*' || symbol == '/') {
    return (2);
  } else if (symbol == '+' || symbol == '-') {
    return (1);
  } else return (0);
}
void InfixToPostfix(char infix_exp[], char postfix_exp[]) {
  int i = 0, j = 0;
  char item, x;
  push('(');
  strcat(infix_exp, ")");
  item = infix_exp[i];
  while (item != '\0') {
    if (item == '(') {
      push(item);
    } else if (isdigit(item) || isalpha(item)) {
      postfix_exp[j] = item;
      j++;
    } else if (is_operator(item) == 1) {
      x = pop();
      while (is_operator(x) == 1 && precedence(x) >= precedence(item)) {
        postfix_exp[j] = x;
        j++;
        x = pop();
      }
      push(x);
      push(item);
    } else if (item == ')') {
      x = pop();
      while (x != '(') {
        postfix_exp[j] = x;
        j++;
        x = pop();
      }
    } else {
      printf("\nInvalid infix Expression.\n");
      getchar();
      exit(1);
    }
    i++;
    item = infix_exp[i];
  }
  if (top > 0) {
    printf("\nInvalid infix Expression.\n");
    getchar();
    exit(1);
  }
  postfix_exp[j] = '$';
}
int main() {
  char infix[SIZE], postfix[SIZE], x, y;
  int i = 0;
  printf("\nEnter Infix expression : ");
  gets(infix);
  InfixToPostfix(infix, postfix);
  printf("Postfix Expression: ");
  puts(postfix);
  while (postfix[i] != '$') {
    if (isalpha(postfix[i])) {
      push(postfix[i]);
    } else if ((postfix[i] == '+') || (postfix[i] == '-') || (postfix[i] == '/') ||
      (postfix[i] == '*') || (postfix[i] == '^')) {
      x = pop();
      y = pop();
      if (x == 't') {
        printf("%c %c t%d t%d\n", postfix[i], y, k, k + 1);
        k++;
      } else if (y == 't') {
        printf("%c t%d %c t%d\n", postfix[i], k, x, k + 1);
        k++;
      } else printf("%c %c %c t%d\n", postfix[i], y, x, k);
      push('t');
    }
    i++;
  }
  return 0;
}

*************************************************************************
OUTPUT
------
Enter Infix expression : a+b*c-d
Postfix Expression: abc*+d-$
* b c t1
+ a t1 t2
- t2 d t3
```

```
Name: ALAN D ANDOOR                                           EXPT NO: 9
DATE: 16/12/22                                                ROLL NO: 6
                                                              BATCH  :S7 CSB
*************************************************************************
                              BACKEND OF COMPILER
                      -------------------

#include<stdio.h>
#include<string.h>
void main(){
        char icode[10][30], str[20], opr[10];
        int i=0;
        printf("\nEnter the set of intermediate code (terminated by exit):\n");
        do{
                scanf("%s", icode[i]);
        }while(strcmp(icode[i++],"exit")!=0);
        printf("\nTarget code generation");
        printf("\n******************");
        i=0;
        do{
                strcpy(str,icode[i]);
                switch(str[3]){
                        case '+':
                                strcpy(opr,"ADD");
                                break;
                        case '-':
                                strcpy(opr,"SUB");
                                break;
                        case '*':
                                strcpy(opr,"MUL");
                                break;
                        case '/':
                                strcpy(opr,"DIV");
                                break;
                }
                printf("\nMov %c,R%d", str[2],i);
                printf("\n%s %c,R%d", opr,str[4],i);
                printf("\nMov R%d,%c", i,str[0]);
        }while(strcmp(icode[++i],"exit")!=0); printf("\n");
}
*************************************************************************
OUTPUT
----------
Enter the set of intermediate code (terminated by exit):
a=a*b
f=q+w
t=q-j
exit

Target Code Generation
************************
Mov a,R0
MUL b,R0
Mov R0,a
Mov q,R1
ADD w,R1
Mov R1,f
Mov q,R2
SUB j,R2
Mov R2,t
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 10
DATE: 22/12/22                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
*************************************************************************
                        VOWELS AND CONSONANTS
                ---------------------

%{
#include<stdio.h>
int v=0;
int c=0;
%}
%%
[\t \n] ;
[aeiouAEIOU] {v++;}
[^aeiouAEIOU] {c++;}
%%
int yywrap()
{
return 1;
}
int main()
{
printf("Enter the string:");
yylex();
printf("\nNo of vowels=%d\nNo of consonants=%d",v,c);
}

*************************************************************************

OUTPUT
------
Enter the string:hello world

No of vowels=3
No of consonants=7
```

```
Name: ALAN D ANDOOR                                           EXPT NO: 11
DATE: 22/12/22                                                ROLL NO: 6
                                                              BATCH  :S7 CSB
*************************************************************************
                 NUMBER OF LINES, WORDS AND CHARACTERS
         -------------------------------------


%{
#include<stdio.h>
#include<string.h>
int l=0,c=0,w=0;
%}
%%
[\n] {l++;c++;}
[a-zA-z]+ {w++;c+=strlen(yytext);}
. {c++;}
%%
int yywrap()
{
return 1;
}
int main()
{
printf("Enter the string:");
yylex();
printf("\nLines=%d\nWords=%d\nCharacters=%d",l,w,c);
}

*************************************************************************

OUTPUT
------
Enter the string:
hello world
welcome to
programming

Lines=3
Words=5
Characters=35
```

```
NameALAN D ANDOOR                                             EXPT NO: 12
DATE: 22/12/22                                                ROLL NO: 6
                                                              BATCH  :S7 CSB
*************************************************************************
                 CONVERT SUBSTRING TO UPPERCASE
         ------------------------------
%{
#include<stdio.h>
#include<string.h>
%}
%%
abc {strcpy(yytext,"ABC");ECHO;}
%%
int yywrap()
{
return 1;
}
int main()
{
printf("Enter the string:");
yylex();
}

*************************************************************************
```

```
OUTPUT
------
Enter the string:theabchgj abc
theABChgj ABC
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 13
DATE: 16/12/22                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
*************************************************************************
                     LEXICAL ANALYZER USING LEX
                     --------------------------

%{
int COMMENT=0;
%}
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* {printf("\n%s is a preprocessor directive",yytext);}
int |
float |
char |
double |
while |
for |
struct |
typedef |
do |
if |
break |
continue |
void |
switch |
return |
else |
goto |
main {printf("\n%s\t is a KEYWORD",yytext);}
"/*" {COMMENT=1;}{printf("\n %s\t is a COMMENT",yytext);}
\(      {if(!COMMENT)printf("\n %s\t FUNCTION ",yytext);}
\{      {if(!COMMENT)printf("\n %s\t IS BLOCK BEGINS",yytext);}
\}      {if(!COMMENT)printf("\n %s\t IS BLOCK ENDS ",yytext);}
\)      {if(!COMMENT)printf("\n %s\t FUNCTION",yytext);}
\;      {if(!COMMENT)printf("\n %s\t SPECIAL CHARACTER",yytext);}
{identifier}(\[[0-9]*\])? {if(!COMMENT) printf("\n %s\t IDENTIFIER",yytext);}
\".*\" {if(!COMMENT)printf("\n%s\t is a STRING",yytext);}
[0-9]+ {if(!COMMENT) printf("\n%s\t is a NUMBER ",yytext);}
= {if(!COMMENT)printf("\n %s\t is an ASSIGNMENT OPERATOR",yytext);}
\<= |
\>= |
\< |
== |
\> {if(!COMMENT) printf("\n%s\t is a RELATIONAL OPERATOR",yytext);}
\+ |
\- |
\* |
\/ {if(!COMMENT) printf("\n%s\t is an ARITHMETIC OPERATOR",yytext);}
%%
int main(int argc, char **argv)
{
FILE *file;
file=fopen("input.c","r");
if(!file)
{
printf("could not open the file");
exit(0);
}
yyin=file;
yylex();
printf("\n");
return(0);
}
int yywrap()
{
return(1);
}
*************************************************************************

INPUT
-----
void main()
{
int a=10;
a=a/1;
}

OUTPUT
------
void     is a KEYWORD
main     is a KEYWORD
 (       FUNCTION
 )       FUNCTION

 {       IS BLOCK BEGINS

int      is a KEYWORD
 a       IDENTIFIER
 =       is an ASSIGNMENT OPERATOR
10       is a NUMBER
 ;       SPECIAL CHARACTER

 a       IDENTIFIER
 =       is an ASSIGNMENT OPERATOR
 a       IDENTIFIER
/        is an ARITHMETIC OPERATOR
1        is a NUMBER
 ;       SPECIAL CHARACTER

 }       IS BLOCK ENDS
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 14
DATE: 06/01/23                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
*************************************************************************
                     RECOGNIZE VALID EXPRESSION USING YACC
                     -------------------------------------


YACC
----
%{
    #include<stdio.h>
    int flag=0;
%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
         //printf("\nResult=%d\n",$$);
         return 0;
        };
E:E'+'E {$$=$1+$3;}
```

```
 |E'-'E {$$=$1-$3;}
 |E'*'E {$$=$1*$3;}
 |E'/'E {$$=$1/$3;}
 |E'%'E {$$=$1%$3;}
 |'('E')' {$$=$2;}
 | NUMBER {$$=$1;}
;
%%
void main()
{
   printf("\nEnter Arithmetic Expression which have operations +, -, *, / and  paranthesis:
   yyparse();
  if(flag==0)
   printf("\nEntered arithmetic expression is Valid\n\n");
}
void yyerror()
{
   printf("\nEntered arithmetic expression is Invalid\n\n");
   flag=1;
}


LEX
---
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}

%%
[0-9]+ {
          yylval=atoi(yytext);
          return NUMBER;
       }
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
**************************************************************************

OUTPUT
------
Enter Arithmetic Expression which have operations +, -, *, / and  paranthesis:
(45/4)+67*2

Entered arithmetic expression is Valid
```

```
Name: ALAN D ANDOOR                                        EXPT NO: 15
DATE: 06/01/23                                             ROLL NO: 6
                                                           BATCH  :S7 CSB
**************************************************************************
                     RECOGNIZE VALID IDENTIFIER USING YACC
                     -------------------------------------


YACC
----
%{
    #include<stdio.h>
    int valid=1;
%}
```

```
%token digit letter

%%
start : letter s
s :     letter s
      | digit s
      |
      ;
%%

int yyerror(){
    printf("\nIts not an identifier!\n");
    valid=0;
    return 0;
}
int main(){
    printf("\nEnter a name to tested for identifier: ");
    yyparse();
    if(valid){
        printf("\nIt is a valid identifier\n");
    }
}


LEX
---
%{
    #include "y.tab.h"
%}

%%
[a-zA-Z_][a-zA-Z_0-9]* return letter;
[0-9]                  return digit;
.                      return yytext[0];
\n                     return 0;
%%

int yywrap(){
        return 1;
}
**************************************************************************

OUTPUT
------
Enter a name to tested for identifier: Dubai_34

It is a identifier
```

```
Name: ALAN D ANDOOR                                         EXPT NO: 16
DATE: 06/01/23                                              ROLL NO: 6
                                                            BATCH  :S7 CSB
**************************************************************************
                     CALCULATOR USING LEX & YACC
                     ---------------------------


LEX
----
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}

%%

[0-9]+ {
```

```
          yylval=atoi(yytext);
          return NUMBER;
       }
[\t] ;
[\n] return 0;
. return yytext[0];

%%

int yywrap(){
return 1;
}


YACC
-----
%{
    #include<stdio.h>
    int flag=0;
%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'

%%
ArithmeticExpression: E{
         printf("\nResult=%d\n",$$);
         return 0;
};
E:E'+'E {$$=$1+$3;}
|E'-'E {$$=$1-$3;}
|E'*'E {$$=$1*$3;}
|E'/'E {$$=$1/$3;}
|E'%'E {$$=$1%$3;}
|'('E')' {$$=$2;}
| NUMBER {$$=$1;}
;
%%

void main(){
   printf("\nEnter Any Arithmetic Expression:\n");
   yyparse();
  if(flag==0)
   printf("\nEntered arithmetic expression is Valid\n\n");
}
void yyerror(){
   printf("\nEntered arithmetic expression is Invalid\n\n");
   flag=1;
}
*************************************************************************

OUTPUT
------
Enter Any Arithmetic Expression:
(5*9)+(36/9)-2

Result=47

Entered arithmetic expression is Valid
```