CD LAB

1️⃣ Lexical Analyzer in C

```
#include <stdio.h>
#include <ctype.h>

int main() {
    char str[100];
    int i;
    printf("Enter input: ");
    scanf("%s", str);

    for(i=0; str[i]!='\0'; i++) {
        if(isalpha(str[i]))
            printf("Identifier: %c\n", str[i]);
        else if(isdigit(str[i]))
            printf("Constant: %c\n", str[i]);
    }
    return 0;
}
```

Input: `a1b2`

Output:

```
Identifier: a
Constant: 1
Identifier: b
Constant: 2
```

3️⃣ Lexical Analyzer using LEX

```
%{
#include <stdio.h>
%}
%%
[0-9]+      printf("Number: %s\n", yytext);
[a-zA-Z]+   printf("Identifier: %s\n", yytext);
"+"|"-"|"="|"*"|"/" printf("Operator: %s\n", yytext);
.           ;
%%
int main() {
  yylex();
  return 0;
}
```

Input: `a=10+5`

Output:

```
Identifier: a
Operator: =
Number: 10
Operator: +
Number: 5
```

4️⃣ YACC – Valid Arithmetic Expression

`expr.y`

```
%{
#include <stdio.h>
int yylex();
void yyerror() { printf("Invalid Expression\n"); }
%}
%token NUM
%%
E : E '+' E | E '-' E | E '*' E | E '/' E | NUM { printf("Valid Expression\n"); }
```

```
 ;
%%
int main() { printf("Enter expression: "); yyparse(); }
```

expr.l

```
%{
#include "y.tab.h"
%}
%%
[0-9]+  return NUM;
[ \t\n]+ ;
.       return yytext[0];
%%
```

Input: 2+3*5

Output: Valid Expression

5️⃣ YACC – Valid Variable Recognition

var.y

```
%{
#include <stdio.h>
int yylex();
void yyerror() { printf("Invalid Variable\n"); }
%}
%token ID
%%
S : ID { printf("Valid Variable\n"); }
 ;
%%
int main() { yyparse(); }
```

var.l

```
%{
#include "y.tab.h"
%}
%%
[a-zA-Z_][a-zA-Z0-9_]* return ID;
. ;
%%
```

Input: `a1`

Output: `Valid Variable`

6️⃣ LEX + YACC – Simple Calculator

`calc.y`

```
%{
#include <stdio.h>
#include <stdlib.h>
int yylex(); void yyerror() {}
%}
%token NUM
%left '+' '-'
%left '*' '/'
%%
S : E { printf("Result = %d\n", $1); }
  ;
E : E '+' E { $$ = $1 + $3; }
  | E '-' E { $$ = $1 - $3; }
  | E '*' E { $$ = $1 * $3; }
  | E '/' E { $$ = $1 / $3; }
  | NUM ;
%%
int main() { printf("Enter expression: "); yyparse(); }
```

`calc.l`

```
%{
#include "y.tab.h"
%}
%%
[0-9]+ { yylval = atoi(yytext); return NUM; }
[ \t\n]+ ;
. return yytext[0];
%%
```

Input: `3+2*4`

Output: `Result = 11`

7️⃣ Three-Address Code Generator

```
#include <stdio.h>
int main() {
    char exp[10];
    printf("Enter expression (a+b*c): ");
    scanf("%s", exp);
    printf("t1 = %c * %c\n", exp[2], exp[4]);
    printf("t2 = %c + t1\n", exp[0]);
    return 0;
}
```

Input: `a+b*c`

Output:

```
t1 = b * c
t2 = a + t1
```

9️⃣ Code Optimization Example

```
#include <stdio.h>
int main() {
    int a = (2 + 3) * 4;
    printf("Optimized: a = %d\n", a);
    return 0;
}
```

Output:

```
Optimized: a = 20
```

1️⃣0️⃣ Machine Code Generation

```
#include <stdio.h>
int main() {
    printf("MOV R1, a\n");
    printf("MOV R2, b\n");
    printf("ADD R1, R2\n");
    printf("MOV c, R1\n");
    return 0;
}
```

Output:

```
MOV R1, a
MOV R2, b
ADD R1, R2
MOV c, R1
```

1️⃣3️⃣ Recognize Control Structures

```
#include <stdio.h>
#include <string.h>
```

```
int main() {
    char str[50];
    printf("Enter control statement: ");
    scanf("%s", str);
    if (strstr(str, "if") || strstr(str, "for") || strstr(str, "while") || strstr(str, "switc
h"))
        printf("Valid Control Structure\n");
    else
        printf("Invalid\n");
    return 0;
}
```

Input: `if(x>0)`

Output: `Valid Control Structure`
