List of Experiments

S. No.	Topic(s)	CO	PO
1	A. Write a simple calculator program in C/C++/JAVA B. Implementation of basic Flex programs	1	1
2	Implementation of Lexical Analyzer using FLEX.	1	2,5
3	Implementation of calculator using FLEX and BISON.	1	2,5
4	Write a program for Elimination of Left recursion/Left factoring in C/C++/JAVA	2	2,5
5	Write a program for to Compute FIRST & FOLLOW for Top-Down Parsing and predictive parsing table in C/C++/JAVA	2	2
6	Write a program for Shift Reduce Parsing in C/C++/JAVA	3	2
7	Write a program for Computation of LEADING AND TRAILING in C/C++/JAVA	3	2
8	Write a program for Computation of LR (0) items in C/C++/JAVA	3	2
9	Write an program for Intermediate code generation as Prefix and Suffix in C/C++/JAVA	4	3
10	Write an program for Intermediate code generation as Quadruple, Triple, Indirect triple in C/C++/JAVA	4	3
11	Write a program to generate machine code for a simple statement in C/C++/JAVA	5	3
12	Implement backpatching in C/C++/JAVA	5	3

EX 2. Implementation of Lexical Analyzer using FLEX.

Procedure:

1. Place the following files in a folder ex2.l and a sample input file (Factorial.c or any input file).
2. flex ex2.l
3. gcc lex.yy.c
4. ./a.out
5. Enter the file name
6. All tokens generated by the parser will be placed in the output

EX2

```
%{
/* need this for the call to atof() below */
#include <math.h>
%}
DIGIT [0-9]
ID [a-z][a-z0-9]*
%%
{DIGIT}+ {printf( "An integer: %s (%d)\n", yytext,atoi( yytext ) );}
{DIGIT}+"."{DIGIT}* {printf( "A float: %s (%g)\n", yytext,atof( yytext ) );}
int|main|return|if|then|begin|end|procedure|function {printf( "A keyword: %s\n", yytext );}
{ID} printf( "An identifier: %s\n", yytext );
"+"|"-"|"*"|"/" printf( "An operator: %s\n", yytext );
"{"[\^{}}\n]*"}" /* eat up one-line comments */
[ \t\n]+ /* eat up whitespace */
,|; {printf("special symbols: %s\n",yytext);}

. printf( "Unrecognized character: %s\n", yytext );
%%
int yywrap(){}
int main( int argc, char **argv )
{
FILE *fp;
char filename[50];
printf("Enter the filename: \n");
scanf("%s",filename);
fp = fopen(filename,"r");
yyin = fp;
yylex();
return 0;
}
```

INPUT.TXT

```
int main()
{
int a,b,c;
a=5;
b=6;
c=a+b;
return 0;
}
```

```
C:\GnuWin32\programs\new>flex ex2.l

C:\GnuWin32\programs\new>gcc lex.yy.c
```

```
C:\GnuWin32\programs\new>a.exe
Enter the filename:
input.txt
A keyword: int
A keyword: main
Unrecognized character: (
Unrecognized character: )
Unrecognized character: {
A keyword: int
An identifier: a
special symbols: ,
An identifier: b
special symbols: ,
An identifier: c
special symbols: ;
An identifier: a
Unrecognized character: =
An integer: 5 (5)
special symbols: ;
An identifier: b
Unrecognized character: =
An integer: 6 (6)
special symbols: ;
An identifier: c
Unrecognized character: =
An identifier: a
An operator: +
An identifier: b
```

```
special symbols: ;
A keyword: return
An integer: 0 (0)
special symbols: ;
Unrecognized character: }
```

TASK TO BE GIVEN TO THE STUDENTS

1. KINDLY PROVIDE EACH STUDENT WITH DIFFERENT INPUT. AMONG THE

FOLLOWING

2. ASK THE STUDENTS TO REWRITE THE CODE BASED ON THE INPUT
3. ASK THEM TO PRINT OUTPUT IN BELOW FORMAT

TOKEN NO	LINE NO	TOKEN	LEXEME
1	1	keyword	int
2	1	keyword	main

EXAMPLE

1. INPUT 1

```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);

    // true if num is perfectly divisible by 2
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```

2. INPUT 2

```c
#include <stdio.h>
int main() {
    int n, i;
    unsigned long long fact = 1;
    printf("Enter an integer: ");
    scanf("%d", &n);

    // shows error if the user enters a negative integer
    if (n < 0)
        printf("Error! Factorial of a negative number doesn't exist.");
    else {
        for (i = 1; i <= n; ++i) {
            fact *= i;
        }
        printf("Factorial of %d = %llu", n, fact);
    }

    return 0;
}
```

3. INPUT 3

```c
#include <stdio.h>

int addNumbers(int n);

int main() {

  int num;
  printf("Enter a positive integer: ");
  scanf("%d", &num);
  printf("Sum = %d", addNumbers(num));
  return 0;
}

int addNumbers(int n) {
  if (n != 0)
    return n + addNumbers(n - 1);
  else
    return n;
}
```

4. INPUT 4

```cpp
#include <iostream>
using namespace std;

// create a class
class Room {

   public:
    double length;
    double breadth;
    double height;

    double calculate_area() {
        return length * breadth;
    }

    double calculate_volume() {
        return length * breadth * height;
    }
};
```

5. INPUT 5

```c
#include <stdio.h>
int main() {
  int a[10][10], transpose[10][10], r, c;
  printf("Enter rows and columns: ");
  scanf("%d %d", &r, &c);

  // asssigning elements to the matrix
  printf("\nEnter matrix elements:\n");
  for (int i = 0; i < r; ++i)
  for (int j = 0; j < c; ++j) {
    printf("Enter element a%d%d: ", i + 1, j + 1);
    scanf("%d", &a[i][j]);
  }

   // computing the transpose
  for (int i = 0; i < r; ++i)
  for (int j = 0; j < c; ++j) {
    transpose[j][i] = a[i][j];
  }


  return 0;
}
```