Exp. No. 4
Design a lexical Analyzer to validate operators to recognize the operators +,-,*,/ using regular arithmetic operators using C
Program:

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

int main() {
   char s[5];

   printf("\nEnter any operator: ");
   fgets(s, sizeof(s), stdin);  // Safe input method
   s[strcspn(s, "\n")] = '\0';  // Remove newline character if present

   switch (s[0]) {
      case '>':
         if (s[1] == '=')
            printf("\nGreater than or equal\n");
         else
            printf("\nGreater than\n");
         break;

      case '<':
         if (s[1] == '=')
            printf("\nLess than or equal\n");
         else
            printf("\nLess than\n");
         break;

      case '=':
         if (s[1] == '=')
            printf("\nEqual to\n");
         else
            printf("\nAssignment\n");
         break;

      case '!':
         if (s[1] == '=')
            printf("\nNot Equal\n");
         else
            printf("\nBitwise NOT\n");
         break;
```

```
        case '&':
            if (s[1] == '&')
                printf("\nLogical AND\n");
            else
                printf("\nBitwise AND\n");
            break;

        case '|':
            if (s[1] == '|')
                printf("\nLogical OR\n");
            else
                printf("\nBitwise OR\n");
            break;

        case '+':
            printf("\nAddition\n");
            break;

        case '-':
            printf("\nSubtraction\n");
            break;

        case '*':
            printf("\nMultiplication\n");
            break;

        case '/':
            printf("\nDivision\n");
            break;

        case '%':
            printf("\nModulus\n");
            break;

        default:
            printf("\nNot an operator\n");
    }

    return 0;
}
```

Output

```
Enter any operator: <=

Less than or equal


=== Code Execution Successful ===
```