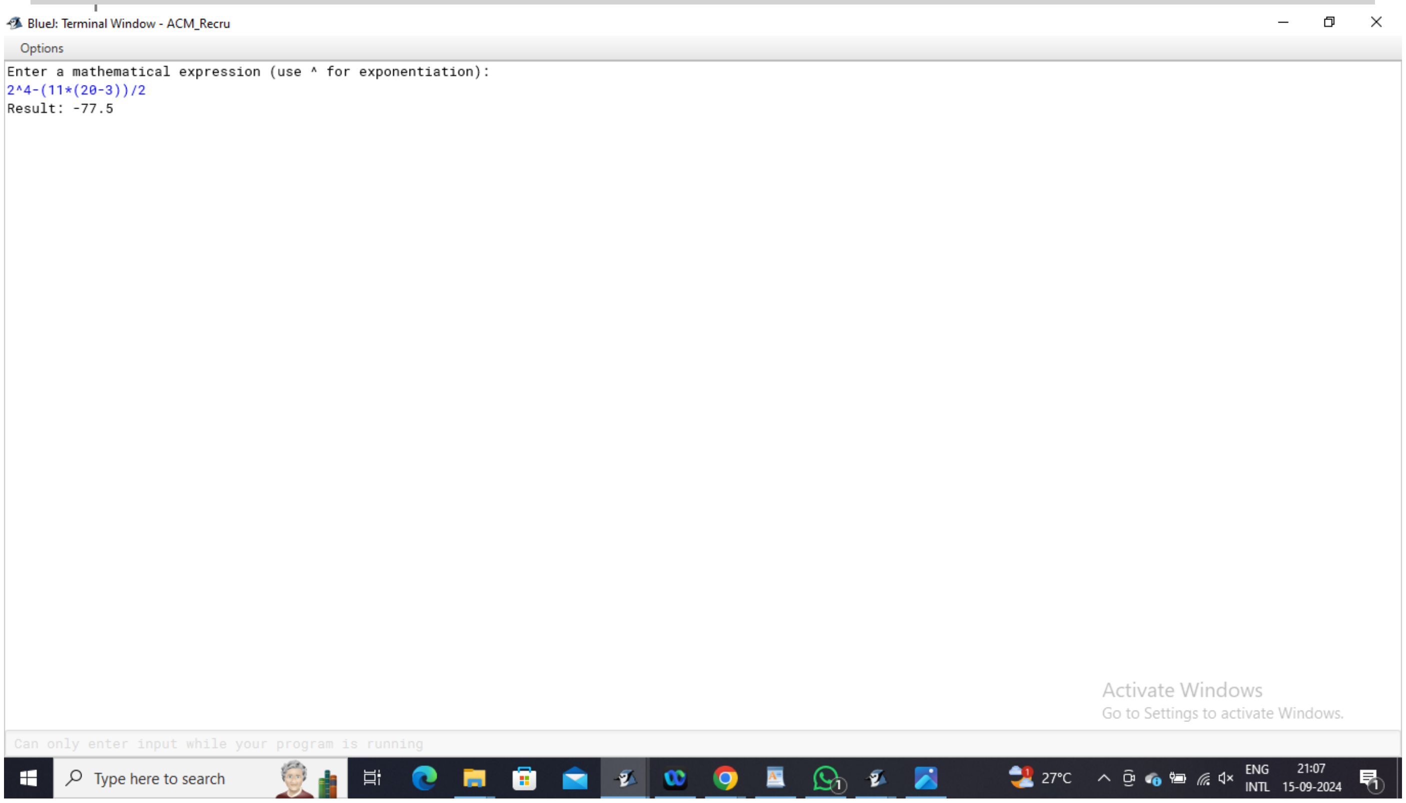
```java
import java.util.*;
class Calculator
 {
    double calculate(double a, double b, char op)
    {                              //performing basic operations
        switch (op)
        {
            case '+': return(a+b);
            case '-': return (a-b);
            case '*': return (a*b);
            case '/': if (b == 0)
                      {
                        System.out.println("Error: Division by zero!");
                        System.exit(0);
                        return 0;
                      }
                      return(a/b);
            case '^': return Math.pow(a,b);
            default : System.out.println("Error: Invalid operator!");
                      System.exit(0);
                      return 0;
        }
    }
    double evaluateExpression(String exp)
    {
    exp = exp.replaceAll(" ", ""); // Removing spaces
    if (exp.contains("("))
    {       // Handling parentheses by solving inner expressions first
        while (exp.contains("("))
        {
            int openIndex = exp.lastIndexOf('(');
            int closeIndex = exp.indexOf(')', openIndex);
            String subExpr = exp.substring(openIndex + 1, closeIndex);
            double subResult = evaluateExpression(subExpr);
            exp = exp.substring(0, openIndex) + subResult + exp.substring(closeIndex
+ 1);
        }
    }
    char[] operators = {'+', '-', '*', '/', '^'};
    for (char operator : operators)
    {
        int index = -1;
        // Handle negative numbers and look for the correct operator position
        if (operator == '+' || operator == '-')
        {
            for (int i = 1; i < exp.length(); i++)
            { // Start from 1 to skip the first character (for negative numbers)
                if (exp.charAt(i) == operator && exp.charAt(i - 1) != '(') {
                    index = i;
                    break;
                }
            }
        }
        else
            index = exp.indexOf(operator);
        if (index != -1)
        {
            String left = exp.substring(0, index);
            String right = exp.substring(index + 1);

            if (operator == '^')
            {
                // For exponentiation, recursively evaluate the right side
                double base = Double.parseDouble(left);
                double exponent = evaluateExpression(right);
                return calculate(base, exponent, operator);
            }
            else
            {
                // For other operators, convert both sides to double
                double leftValue = evaluateExpression(left);
                double rightValue = evaluateExpression(right); // Use evaluateExpression for right side
                return calculate(leftValue, rightValue, operator);
            }
        }
     }
    // If no operator found, it's just a number, so return it as a double
    return Double.parseDouble(exp);
   }
    void main()
    {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter a mathematical expression (use ^ for exponentiation): ");
        String expression = in.nextLine();
        double result = evaluateExpression(expression);
        System.out.println("Result: " + result);
    }
}
```


BlueJ: Terminal Window - ACM_Recru
Options
Activate Windows
Go to Settings to activate Windows.
Can only enter input while your program is running
Type here to search
27°C
ENG
INTL
21:07
15-09-2024

```
Enter a mathematical expression (use ^ for exponentiation):
2^4-(11*(20-3))/2
Result: -77.5
```