***EXPERIMENT 1***

***AIM***: To Convert an Infix expression to a postfix expression

***REQUIREMENTS:*** Visual Studio Code, Java Compiler.

***CODE***:

import java.util.Stack;

public class InfixToPostfix {

    public static int precedence(char operator) {

        switch (operator) {

            case '+':

            case '-':

                return 1;

            case '\*':

            case '/':

                return 2;

            case '^':

                return 3;

        }

        return -1;

    }

    public static String infixToPostfix(String expression) {

        StringBuilder result = new StringBuilder();

        Stack<Character> stack = new Stack<>();

        for (int i = 0; i < expression.length(); i++) {

            char c = expression.charAt(i);

            if (Character.isLetterOrDigit(c)) {

                result.append(c);

            } else if (c == '(') {

                stack.push(c);

            } else if (c == ')') {

                while (!stack.isEmpty() && stack.peek() != '(') {

                    result.append(stack.pop());

                }

                if (!stack.isEmpty() && stack.peek() != '(') {

                    return "Invalid Expression";

                } else {

                    stack.pop();

                }

            } else {

                while (!stack.isEmpty() && precedence(c) <= precedence(stack.peek())) {

                    result.append(stack.pop());

                }

                stack.push(c);

            }

        }

        while (!stack.isEmpty()) {

            if (stack.peek() == '(') {

                return "Invalid Expression";

            }

            result.append(stack.pop());

        }

        return result.toString();

    }

    public static void main(String[] args) {

        String infixExpression = "a+b\*c/d-e";

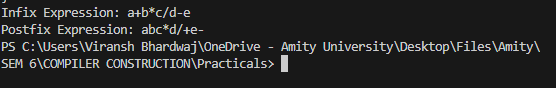
        System.out.println("Infix Expression: " + infixExpression);

        System.out.println("Postfix Expression: " + infixToPostfix(infixExpression));

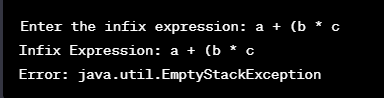
    }

}

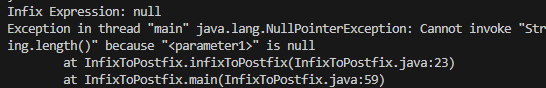
***OUTPUT:***



***ERRORS:***

 A black background with white text

Description automatically generated



|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) | | |

***EXPERIMENT 2***

***AIM***: To read the contents of a file and return them

***REQUIREMENTS:*** Visual Studio Code, Java Compiler.

***CODE:***

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class ReadFile {

    public static void main(String[] args) {

        try {

            // Change "file.txt" to the path of your text file

            BufferedReader reader = new BufferedReader(new FileReader("file.txt"));

            String line;

            while ((line = reader.readLine()) != null) {

                System.out.println(line);

            }

            reader.close();

        } catch (IOException e) {

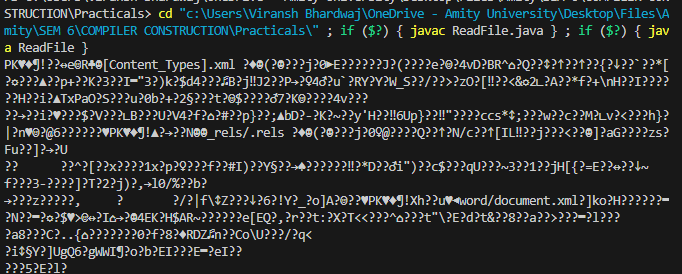
            System.out.println("An error occurred: " + e.getMessage());

        }

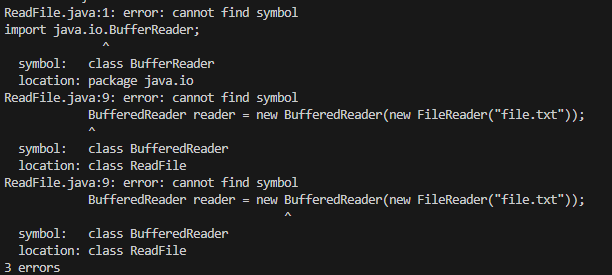
    }

}

***OUTPUT:***



***ERRORS:***



A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) | | |

***EXPERIMENT 3***

***AIM***: To Count the Number Of Tokens In Another File.

***REQUIREMENTS***: Visual Studio Code, Java Compiler.

***CODE:***

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

public class Tokens {

    public static void main(String[] args) throws IOException {

        String path = "Random.java";

        BufferedReader reader = new BufferedReader(new FileReader(path));

        String line;

        int counter = 0;

        while ((line = reader.readLine()) != null) {

            // System.out.println(line);

            char x = '\n';

            for (int i = 0; i < line.length(); i++) {

                char c = line.charAt(i);

                if (!(Character.isLetter(c) || Character.isDigit(c) || c == ' ' || c == '\n')) {

                    counter += 1;

                }

            }

        }

        reader.close();

        System.out.println(counter);

    }

}

***RANDOM FILE:***

public class Random {

    public static void main(String[] args) {

        System.out.println("Hello, World!");

    }

}

***OUTPUT:***

A black background with green and blue text

Description automatically generated

***ERRORS:***

A screen shot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Total Marks** | **Marks Obtained** | **Comments** |
| Concept (A) | 2 |  |  |
| Implementation (B) | 2 |  |  |
| Performance (C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) | | |