INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
SONEPAT

Compiler Design Lab

(CSC 506)

Submitted By:-

Dipankar Yadav

12111070
Branch: CSE
Semester: V
Session: 2021-25

Submitted To:-

Dr. Chanchal Kumar

Index

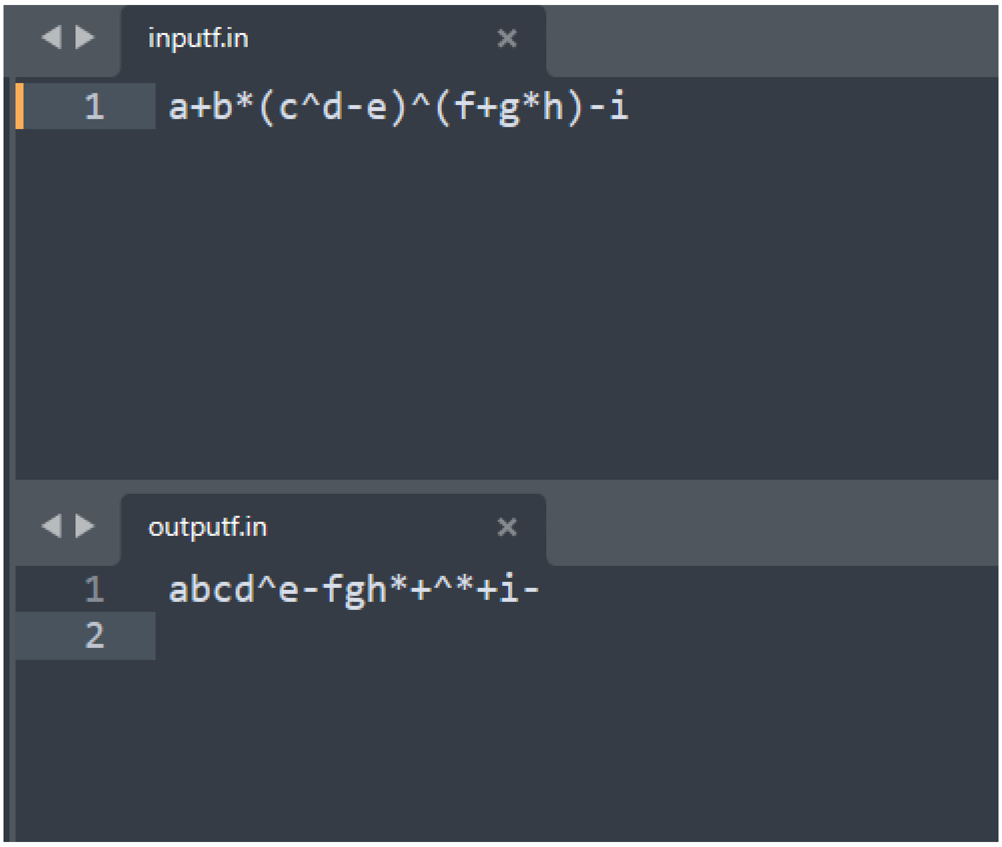
Program-1

Q) Conversion of infix to postfix notation

Code:-

```
#include <bits/stdc++.h>
using namespace std;

int prec(char c) {
    if (c == '^') return 3;
    else if (c == '/' || c == '*') return 2;
    else if (c == '+' || c == '-') return 1;
    else return -1;
}

void infixToPostfix(string s) {
    stack<char> st;
    string result;
    for(int i=0; i<s.length(); i++) {
        char c = s[i];
      if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= '0' && c <= '9')) result += c;
        else if (c == '(') st.push('(');
        else if (c == ')') {
            while (st.top() != '(') {
                result += st.top();
                st.pop();
            }
            st.pop();
        }
        else {
            while (!st.empty() && (prec(s[i]) <= prec(st.top()))) {
                result += st.top();
                st.pop();
            }
            st.push(c);
        }
    }

    while (!st.empty()) {
        result += st.top();
        st.pop();
    }

    cout << result << endl;
}

int main() {
    string exp = "a+b*(c^d-e)^(f+g*h)-i";
    infixToPostfix(exp);
    return 0;
}
```

<u>Output</u>:-



```
a+b*(c^d-e)^(f+g*h)-i
```

```
abcd^e-fgh*+^*+i-
```

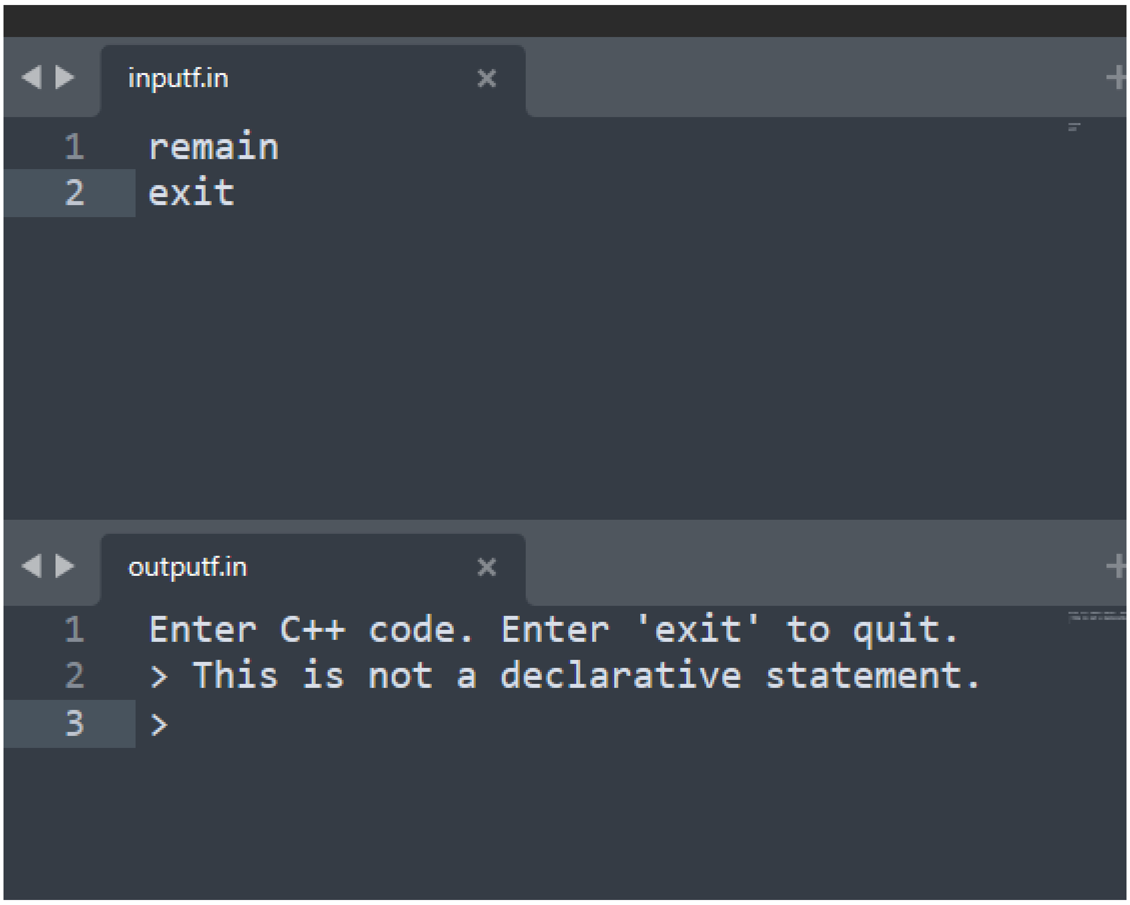
Program-2

Q) Recognize declarative statements.

Code:-

```
#include <iostream>

#include <string>

#include <regex>

using namespace std;


bool isDeclarativeStatement(const string &line) {

    regex varDeclaration("\\s*\\w+\\s+\\w+\\s*;");

    regex funcDeclaration("\\s*\\w+\\s+\\w+\\s*\\(.*\\)\\s*;");

    void myFunction();

    return regex_match(line, varDeclaration) || regex_match(line,
funcDeclaration);

}


int main() {

    cout << "Enter C++ code. Enter 'exit' to quit." << endl;

    string line;

    while (true) {

        cout << "> ";

        getline(cin, line);

        if(line == "exit") {

            break;

        }

        if(isDeclarativeStatement(line)) {

            cout << "This is a declarative statement." << endl;

        }

        else {

            cout << "This is not a declarative statement." << endl;
        }
    }

    return 0;
}
```

<u>Output</u>:-



inputf.in

```
remain
exit
```

outputf.in

```
Enter C++ code. Enter 'exit' to quit.
> This is not a declarative statement.
>
```

Program-3

Q) Program to recognize arithmetic expressions.

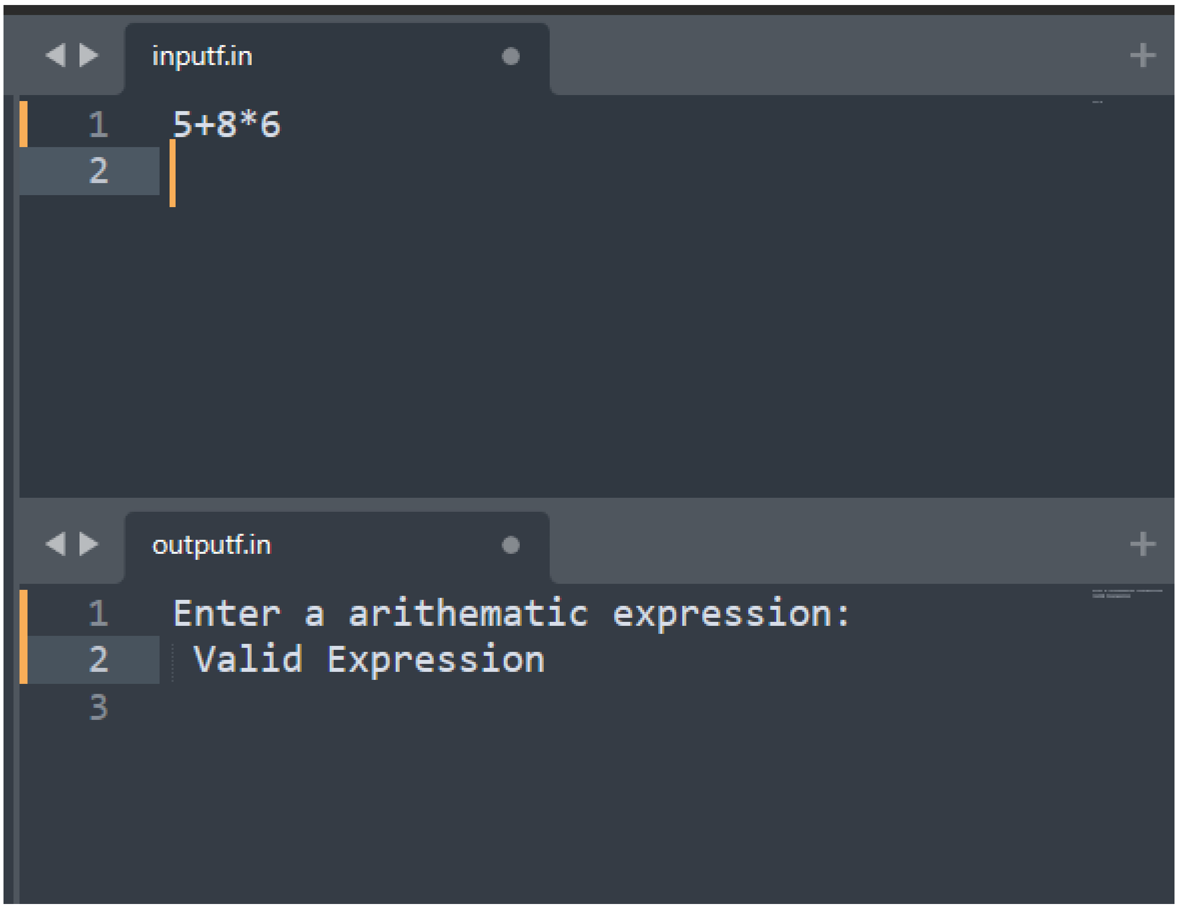
Code:-

```
#include<bits/stdc++.h>
using namespace std;

bool isDigit(char ch) {
    return (ch >= '0' && ch <= '9');
}

bool isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}

bool isBracketOpen(char ch) {
    return (ch == '(' || ch == '[' || ch == '{');
}

char getCorrespondingChar(char ch) {
    if(ch == ')') {
        return '(';
    }
    else if(ch == ']') {
        return '[';
    }
    return '{';
}

bool isValid(string str) {
    int n = str.size();
    stack<int> st1;
```

```
stack<char> st2;
bool result = true;
bool isTrue = true;

for(int i=0; i<n; i++) {
    char temp = str[i];
    if(isDigit(temp)) {
        st1.push(temp - '0');
        if(isTrue) {
            isTrue = false;
        }
        else return false;
    }
    else if(isOperator(temp)) {
        st2.push(temp);
        isTrue = true;
    }
    else {
        if(isBracketOpen(temp)) {
            st2.push(temp);
        }
        else {
            bool flag = true;
            while (!st2.empty()) {
                char c = st2.top();
                st2.pop();
                if(c == getCorrespondingChar(temp)) {
                    flag = false;
                    break;
                }
                else {
                    if(st1.size() < 2) {
                        return false;
                    }
```

```
                        else {
                            st1.pop();
                        }
                    }
                }
                if(flag) {
                    return false;
                }
            }
        }
    }

    while (!st2.empty()) {
        char c = st2.top();
        st2.pop();
        if(!isOperator(c)) {
            return false;
        }
        if(st1.size() < 2) {
            return false;
        }
        else {
            st1.pop();
        }
    }

    if(st1.size() > 1 || !st2.empty()) {
        return false;
    }

    return result;
}

int main() {
    string str;
    cout<<"Enter a arithematic expression: ";
    cin>>str;

    if(isValid(str)) {
        cout<<"Valid Expression\n";
    }
    else cout<<"Not a valid expression\n";
    return 0;
}
```

<u>Output</u>:-



```
5+8*6
```

```
Enter a arithematic expression:
 Valid Expression
```

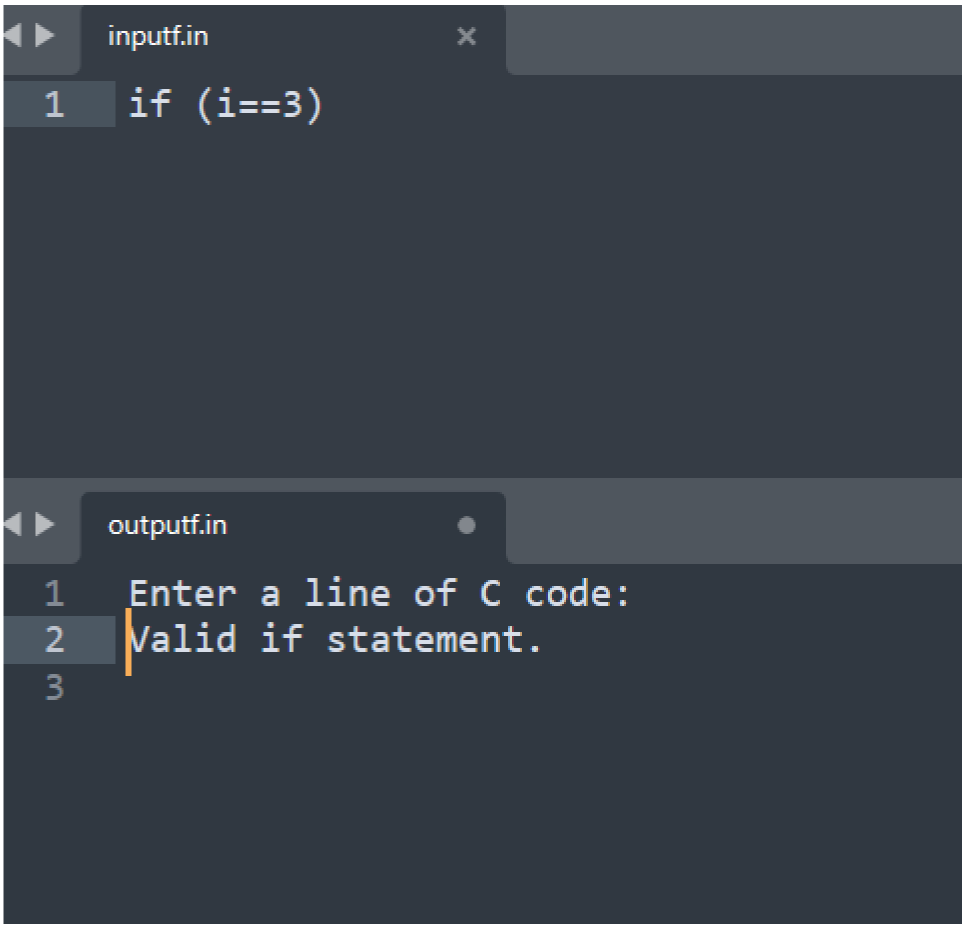
Program-4

Q) Program to check valid IF statements in C and report errors to users.

Code:-

```
#include <stdio.h>

#include <stdbool.h>

#include <string.h>


bool isValidIfStatement(const char *line) {

    int length = strlen(line);

    bool foundIfKeyword = false;

    bool foundOpeningBrace = false;

    bool foundClosingBrace = false;


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

        char c = line[i];

        if(c == ' ' || c == '\t') {

            continue;

        }


        if(!foundIfKeyword) {

            if(c == 'i' && line[i + 1] == 'f' && (line[i + 2] == ' ' || line[i + 2] == '(')) {

                foundIfKeyword = true;

                i += 2;

                continue;

            }

            else {

                return false;

            }

        }

        if(foundIfKeyword && !foundOpeningBrace) {

            if(c == '(') {
                foundOpeningBrace = true;
                continue;
            }
            else {
                return false;
            }
        }
        if(foundIfKeyword && foundOpeningBrace && !foundClosingBrace) {
            if (c == ')') {
                foundClosingBrace = true;
                continue;
            }
        }
    }

    return foundIfKeyword && foundOpeningBrace && foundClosingBrace;
}

int main() {
    char line[100];
    printf("Enter a line of C code: ");
    fgets(line, sizeof(line), stdin);

    if (isValidIfStatement(line)) {
        printf("Valid if statement.\n");
    }
    else {
        printf("Invalid if statement.\n");
    }
    return 0;
}
```

<u>Output</u>:-



```
if (i==3)
```

```
Enter a line of C code:
Valid if statement.
```

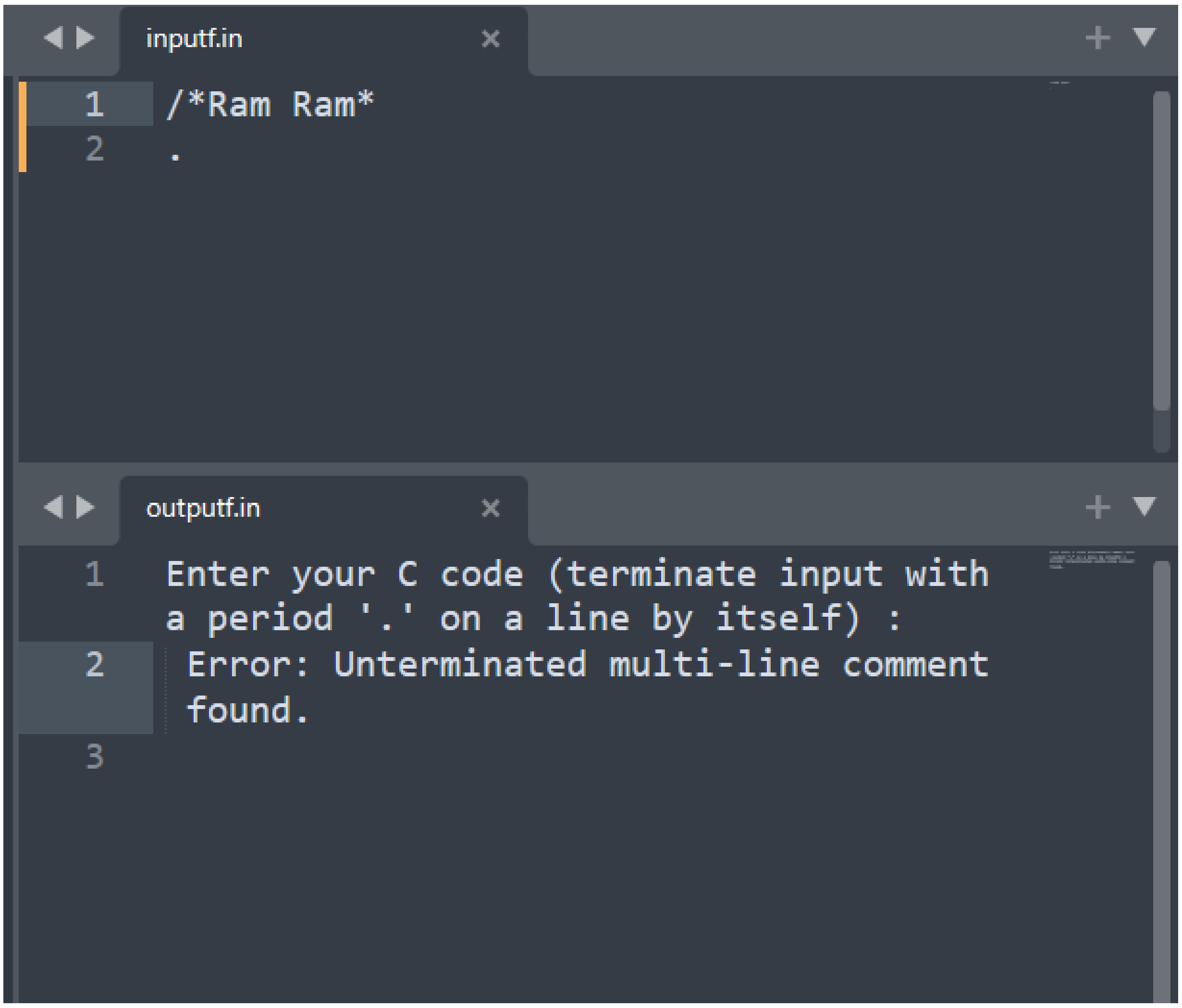
Program-5

Q) Program to check for unterminated, multi-line comment statements in C program.

Code:-

```
#include <stdio.h>

#include <stdbool.h>

#include <string.h>


bool hasUnterminatedComment(const char *code) {

    bool inComment = false;

    int length = strlen(code);

    for(int i=0; i<length-1; i++) {

        if(code[i] == '/' && code[i + 1] == '*') {

            inComment = true;

            i++;

        }

        else if(code[i] == '*' && code[i + 1] == '/') {

            inComment = false;

            i++;

        }

    }

    return inComment;

}


int main() {

    char code[1000];

    printf("Enter your C code (terminate input with a period '.' on a line by itself)
:\n ");


    while(1) {

        char line[100];

        fgets(line, sizeof(line), stdin);
        if(line[0] == '.' && line[1] == '\n') {
            break;
        }
        strcat(code, line);
    }

    if(hasUnterminatedComment(code)) {
        printf("Error: Unterminated multi-line comment found.\n");
    }
    else {
        printf("No unterminated multi-line comments found.\n");
    }

    return 0;
}
```

<u>Output</u>:-



```
/*Ram Ram*
.
```

```
Enter your C code (terminate input with
a period '.' on a line by itself) :
 Error: Unterminated multi-line comment
 found.
```

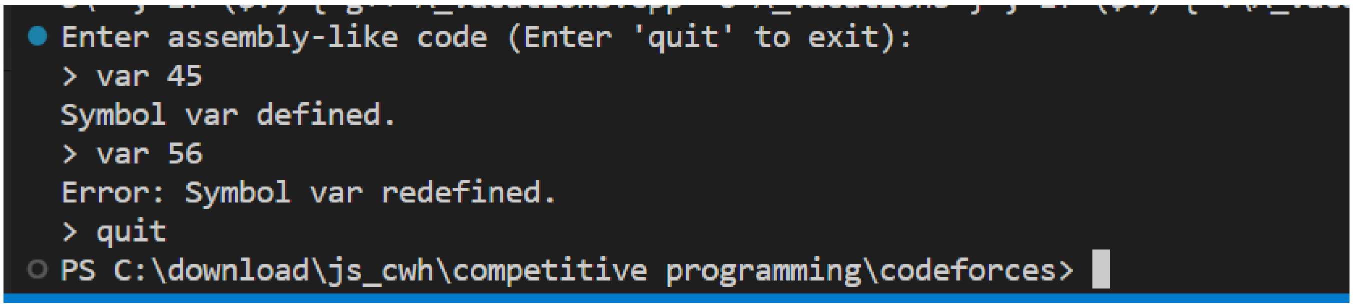
Program-6

Q) Create assembler that will display errors when symbols are used but not defined and vice versa.

Code:-

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX_SYMBOLS 100


typedef struct {

    char name[20];

    int value;

    int defined;

} Symbol;


Symbol symbolTable[MAX_SYMBOLS];

int symbolCount = 0;


void addSymbol(const char *name, int value) {

    if(symbolCount < MAX_SYMBOLS) {

        strcpy(symbolTable[symbolCount].name, name);

        symbolTable[symbolCount].value = value;

        symbolTable[symbolCount].defined = 1;

        symbolCount++;

    }

    else {

        printf("Error: Symbol table is full.\n");

        exit(1);

    }

}


int findSymbol(const char *name) {
    for(int i=0; i<symbolCount; i++) {
        if (strcmp(symbolTable[i].name, name) == 0) {
            return i;
        }
    }
    return -1;
}

int main() {
    char input[100];
    char symbolName[20];
    int symbolValue;
    printf("Enter assembly-like code (Enter 'quit' to exit):\n");
    while(1) {
        printf("> ");
        fgets(input, sizeof(input), stdin);
        if(strcmp(input, "quit\n") == 0) {
            break;
        }

        if(sscanf(input, "%s %d", symbolName, &symbolValue) == 2) {
            int symbolIndex = findSymbol(symbolName);
            if(symbolIndex == -1) {
                addSymbol(symbolName, symbolValue);
                printf("Symbol %s defined.\n", symbolName);
            }
            else {
                printf("Error: Symbol %s redefined.\n", symbolName);
            }
        }
        else {
            int symbolIndex = findSymbol(input);
            if(symbolIndex == -1) {
                printf("Error: Symbol %s used but not defined.\n", input);
            }
            else {
                printf("Symbol %s has a value of %d.\n", input,
symbolTable[symbolIndex].value);
            }
        }
    }

    return 0;
}
```

Output:-



```
● Enter assembly-like code (Enter 'quit' to exit):
  > var 45
  Symbol var defined.
  > var 56
  Error: Symbol var redefined.
  > quit
○ PS C:\download\js_cwh\competitive programming\codeforces> 
```

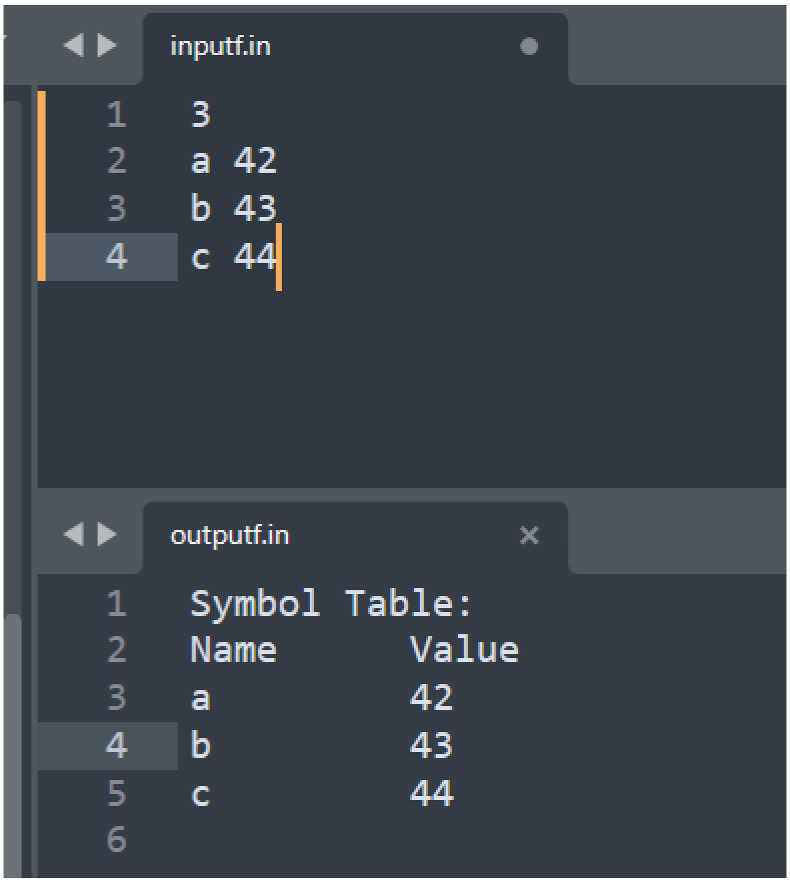
Program-7

Q) Program to create and display content of Symbol table.

Code:-

```
#include <stdio.h>
#include <string.h>
#define MAX_SYMBOLS 100

typedef struct {
    char name[20];
    int value;
} Symbol;

Symbol symbolTable[MAX_SYMBOLS];
int symbolCount = 0;

void addSymbol(const char *name, int value) {
    if(symbolCount < MAX_SYMBOLS) {
        strcpy(symbolTable[symbolCount].name, name);
        symbolTable[symbolCount].value = value;
        symbolCount++;
    }
    else {
        printf("Error: Symbol table is full.\n");
    }
}

void displaySymbolTable() {
    printf("Symbol Table:\n");
    printf("%-10s%-10s\n", "Name", "Value");
    for(int i=0; i<symbolCount; i++) {
        printf("%-10s%-10d\n", symbolTable[i].name, symbolTable[i].value);
    }
}

int main() {
    addSymbol("x", 42);
    addSymbol("y", 100);
    addSymbol("z", 75);
    displaySymbolTable();
    return 0;
}
```

Output:-


inputf.in

```
3
a 42
b 43
c 44
```

outputf.in

```
Symbol Table:
Name        Value
a           42
b           43
c           44

```

Program-8

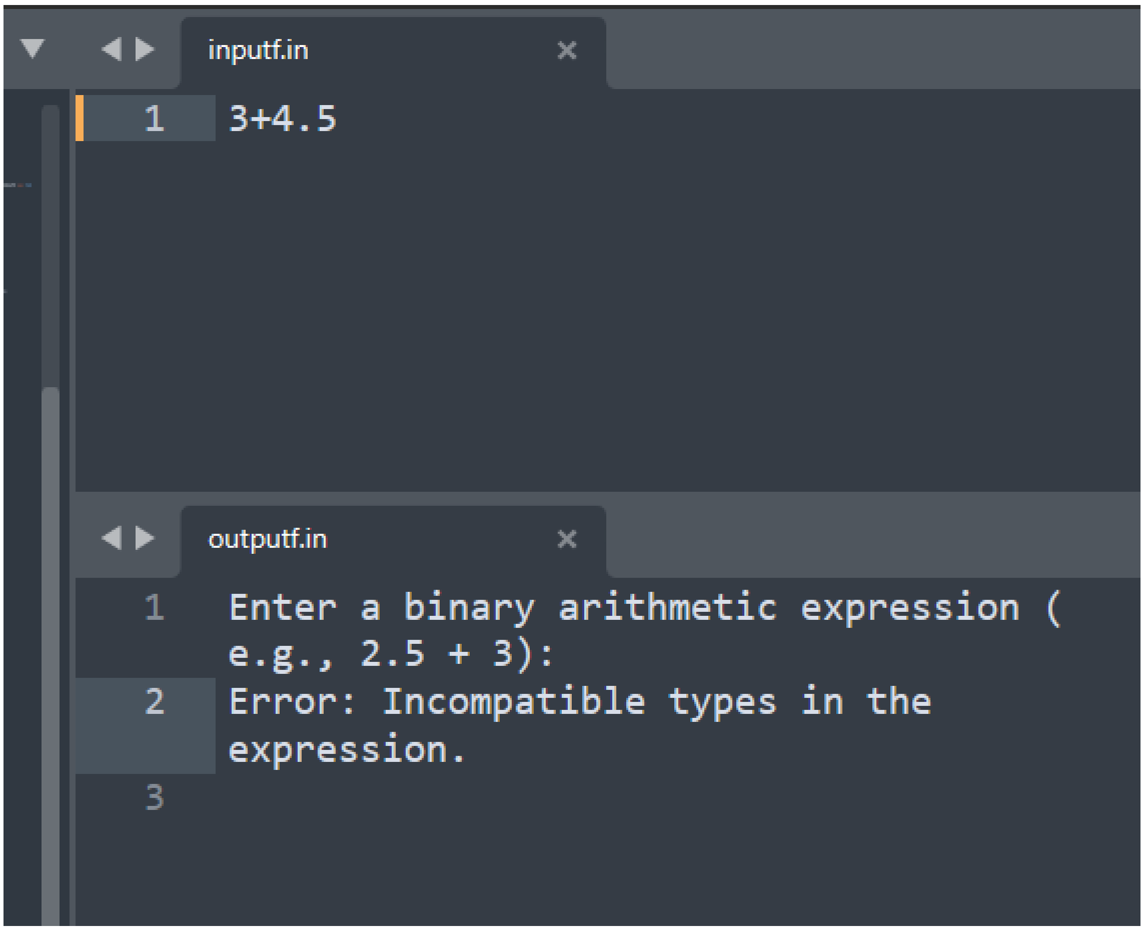
Q) Program to implement type checking.

Code:-

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

typedef enum {
    INT,
    FLOAT,
    ERROR
} DataType;

DataType checkBinaryOperation(DataType left, char operator, DataType right) {
    switch(operator) {
        case '+':
        case '-':
        case '*':
        case '/':
            if(left == INT && right == INT) {
                return INT;
            }
            else if((left == INT && right == FLOAT) || (left == FLOAT && right == INT)
|| (left == FLOAT && right == FLOAT)) {
                return FLOAT;
            }
            else {
                return ERROR;
            }
        default:
            return ERROR;
    }
}
```

```
int main() {
    char operator;
    DataType leftType, rightType, resultType;
    printf("Enter a binary arithmetic expression (e.g., 2.5 + 3):\n");
    scanf("%lf %c %lf", &leftType, &operator, &rightType);
    resultType = checkBinaryOperation(leftType, operator, rightType);

    if(resultType == ERROR) {
        printf("Error: Incompatible types in the expression.\n");
    }
    else {
        printf("Result type of the expression: %s\n", resultType == INT ? "int" :
"float");
    }

    return 0;
}
```

<u>Output</u>:-



```
3+4.5
```

```
Enter a binary arithmetic expression (
e.g., 2.5 + 3):
Error: Incompatible types in the
expression.
```