LAB 5

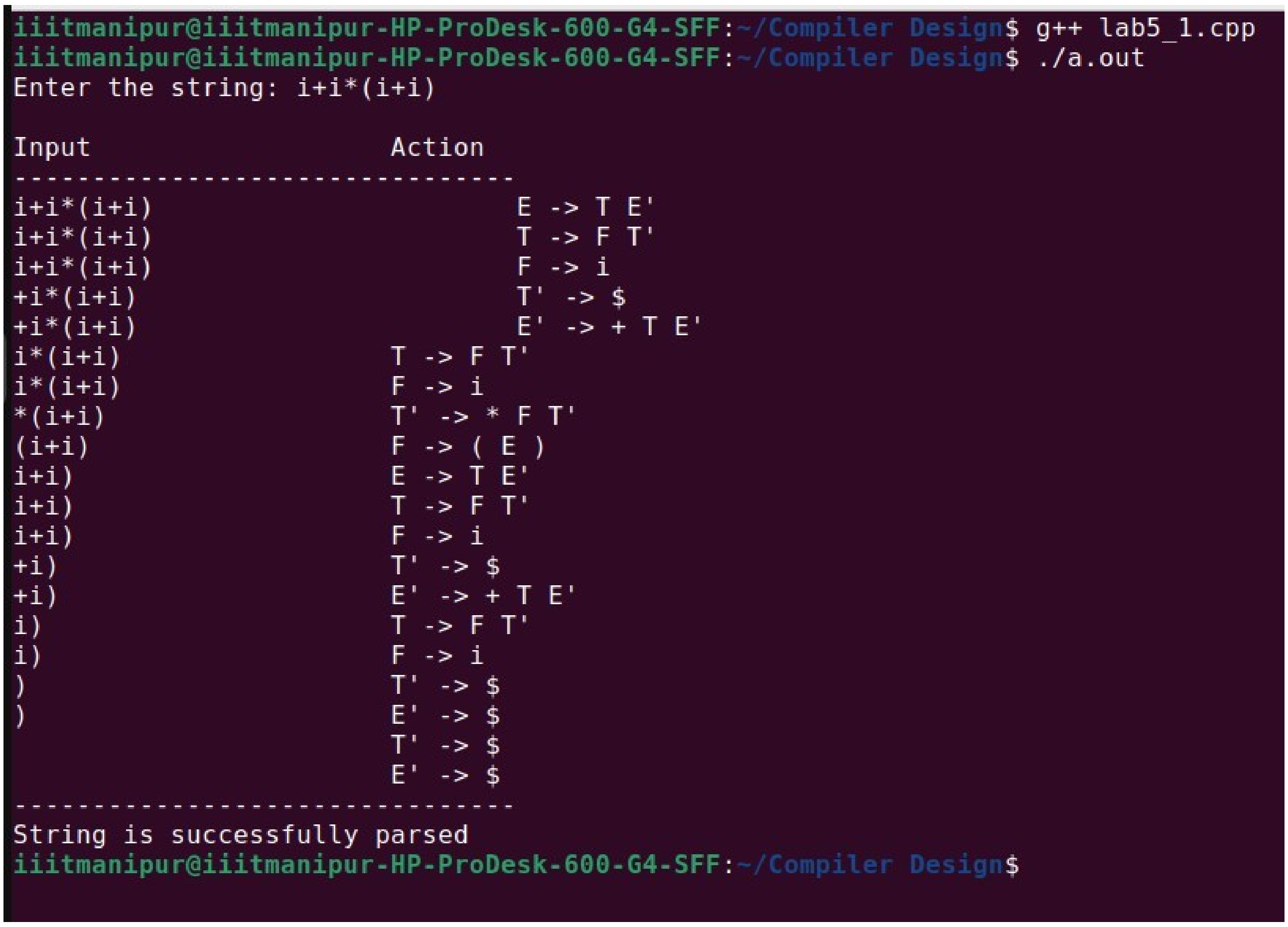
Name :- Amit Kumar
Roll No. :- 220103021
Section :- B

1. Write a program to implement recursive descent parser.

```
#include <iostream>
#include <string>

#define SUCCESS 1
#define FAILED 0
using namespace std;

// Function prototypes
int E(), Edash(), T(), Tdash(), F();

const char *cursor;
string inputString;

int main() {
    cout << "Enter the string: ";
    cin >> inputString; // Read input from the user
    cursor = inputString.c_str();

    cout << endl << "Input\t\t\tAction" << endl;
    cout << "--------------------------------" << endl;

    // Call the starting non-terminal E
    if (E() && *cursor == '\0') { // If parsing is successful and the cursor has reached the
end
        cout << "--------------------------------" << endl;
        cout << "String is successfully parsed" << endl;
        return 0;
    } else {
        cout << "--------------------------------" << endl;
        cout << "Error in parsing string" << endl;
        return 1;
    }
}

// Grammar rule: E -> T E'
int E() {
    cout << cursor << "\t\t\tE -> T E'" << endl;
    if (T()) { // Call non-terminal T
        if (Edash()) // Call non-terminal E'
            return SUCCESS;
        else
            return FAILED;
    } else
        return FAILED;
```

```
}

// Grammar rule: E' -> + T E' | $
int Edash() {
    if (*cursor == '+') {
        cout << cursor << "\t\t\tE' -> + T E'" << endl;
        cursor++;
        if (T()) { // Call non-terminal T
            if (Edash()) // Call non-terminal E'
                return SUCCESS;
            else
                return FAILED;
        } else
            return FAILED;
    } else {
        cout << cursor << "\t\t\tE' -> $" << endl;
        return SUCCESS;
    }
}

// Grammar rule: T -> F T'
int T() {
    cout << cursor << "\t\t\tT -> F T'" << endl;
    if (F()) { // Call non-terminal F
        if (Tdash()) // Call non-terminal T'
            return SUCCESS;
        else
            return FAILED;
    } else
        return FAILED;
}

// Grammar rule: T' -> * F T' | $
int Tdash() {
    if (*cursor == '*') {
        cout << cursor << "\t\t\tT' -> * F T'" << endl;
        cursor++;
        if (F()) { // Call non-terminal F
            if (Tdash()) // Call non-terminal T'
                return SUCCESS;
            else
                return FAILED;
        } else
            return FAILED;
    } else {
        cout << cursor << "\t\t\tT' -> $" << endl;
        return SUCCESS;
    }
}

// Grammar rule: F -> ( E ) | i
int F() {
    if (*cursor == '(') {
        cout << cursor << "\t\t\tF -> ( E )" << endl;
        cursor++;
        if (E()) { // Call non-terminal E
```

```
            if (*cursor == ')') {
                cursor++;
                return SUCCESS;
            } else
                return FAILED;
        } else
            return FAILED;
    } else if (*cursor == 'i') {
        cout << cursor << "\t\t\tF -> i" << endl;
        cursor++;
        return SUCCESS;
    } else
        return FAILED;
}
```



```
iiitmanipur@iiitmanipur-HP-ProDesk-600-G4-SFF:~/Compiler Design$ g++ lab5_1.cpp
iiitmanipur@iiitmanipur-HP-ProDesk-600-G4-SFF:~/Compiler Design$ ./a.out
Enter the string: i+i*(i+i)

Input                   Action
--------------------------------
i+i*(i+i)                       E -> T E'
i+i*(i+i)                       T -> F T'
i+i*(i+i)                       F -> i
+i*(i+i)                        T' -> $
+i*(i+i)                        E' -> + T E'
i*(i+i)                 T -> F T'
i*(i+i)                 F -> i
*(i+i)                  T' -> * F T'
(i+i)                   F -> ( E )
i+i)                    E -> T E'
i+i)                    T -> F T'
i+i)                    F -> i
+i)                     T' -> $
+i)                     E' -> + T E'
i)                      T -> F T'
i)                      F -> i
)                       T' -> $
)                       E' -> $
                        T' -> $
                        E' -> $
--------------------------------
String is successfully parsed
iiitmanipur@iiitmanipur-HP-ProDesk-600-G4-SFF:~/Compiler Design$
```