# Compiler Design Lab

**Date:** 18-12-2025  **Name:** S Vishwajith
**Assignment No.:** 3  **Register No.:** 23BCE1145

## Aim:

To write a C++ code to find Nullable(), Firstpos(), Lastpos() and Followpos() for the given Regular Expressions: (a|b)*a(a|b)(a|b), (a|b)*a(a|b)*, (a|b)*|(ac)*.

## Algorithm:

1. Concatenate the Regular Expression with '#' for the end marker.
2. Construct a syntax tree.
3. Calculate the Nullable(), Firstpos(), Lastpos() and Followpos() using the rules for leaf nodes, Star (*) nodes, Concatenation (.) nodes and Or (|) nodes.
4. Maintain a vector of the number of characters that are not operators and print their Followpos().

## Code:

### RE_to_DFA.cpp:

```
#include <iostream>

#include <vector>

#include <string>

#include <set>

#include <stack>

#include <algorithm>

using namespace std;


string infix_to_postfix(string infix)

{

  string pf = "";

  stack<char> op_stack;
```

```
for (char c : infix)
{
  if (c != '*' && c != '|' && c != '.' && c != '(' && c != ')')
  {
    pf += c;
  }
  else if (c == '(')
  {
    op_stack.push(c);
  }
  else if (c == ')')
  {
    while (!op_stack.empty() && op_stack.top() != '(')
    {
      pf += op_stack.top();
      op_stack.pop();
    }
    op_stack.pop();
  }
  else
  {
    while (!op_stack.empty() &&
        ((c == '*' && (op_stack.top() == '*')) ||
         (c == '.' && (op_stack.top() == '*' || op_stack.top() == '.')) ||
         (c == '|' && (op_stack.top() == '*' || op_stack.top() == '.' || op_stack.top() == '|'))))
    {
      pf += op_stack.top();
      op_stack.pop();
```

```
            }
            op_stack.push(c);
        }
    }
    while (!op_stack.empty())
    {
        pf += op_stack.top();
        op_stack.pop();
    }
    return pf;
}

vector<int> operator+(vector<int> a, vector<int> b)
{
    a.insert(a.end(), b.begin(), b.end());
    sort(a.begin(), a.end());
    a.erase(unique(a.begin(), a.end()), a.end());
    return a;
}

class tree
{
public:
    char c;
    int lno;
    vector<int> first, last;
    bool nullable;
    tree *left;
    tree *right;

    tree(char ch, int n)
    {
        c = ch;
        lno = n;
        nullable = false;
        left = nullptr;
        right = nullptr;
        first.clear();
        last.clear();
    }
};

int main()
{
    string rx;
    cout << "Enter a Regular Expression in infix form: ";
    cin >> rx;
    rx = "(" + rx + ").#";
    string postfix = infix_to_postfix(rx);
    cout << "Postfix Expression: " << postfix << endl
         << endl;

    vector<vector<int>> follow;
    vector<char> chars;
    set<char> inputs;
    stack<tree *> st;
```

```
  int leafno = 0;
  tree *temp;

  for (int i = 0; i < postfix.size(); i++)
  {
    char t = postfix[i];
    if (t != '*' && t != '.' && t != '|')
    {
      cout << "Character at leaf node number " << leafno << " is: " << t << ". ";
      chars.push_back(t);
      if (t != '#')
        inputs.insert(t);
      temp = new tree(t, leafno);
      temp->first.push_back(leafno);
      temp->last.push_back(leafno);

      follow.push_back(vector<int>());
      cout << "This node isn't nullable." << endl;
      cout << "Firstpos: {" << leafno << "}." << endl;
      cout << "Lastpos: {" << leafno << "}." << endl
           << endl;

      leafno++;
    }
    else if (t == '*')
    {
      cout << "This isn't a leaf node. It contains the character: '*' (star). It is nullable." <<
endl;
```

```
    temp = new tree(t, -1);
    temp->left = st.top();
    st.pop();
    temp->first = temp->left->first;
    temp->last = temp->left->last;
    temp->nullable = true;
    cout << "Firstpos: {";
    for (int j = 0; j < temp->first.size(); j++)
    {
        cout << temp->first[j] << ", ";
    }
    cout << "}." << endl;
    cout << "Lastpos: {";
    for (int j = 0; j < temp->last.size(); j++)
    {
        cout << temp->last[j] << ", ";
        follow[temp->last[j]] = follow[temp->last[j]] + temp->first;
    }
    cout << "}." << endl
         << endl;
}
else if (t == '.')
{
    cout << "This isn't a leaf node. It contains the character: '.' (concatenation). ";
    temp = new tree(t, -1);
    temp->right = st.top();
    st.pop();
    temp->left = st.top();
```

```
st.pop();
temp->nullable = temp->left->nullable && temp->right->nullable;
if (temp->nullable)
   cout << "The node is nullable." << endl;
else
   cout << "The node isn't nullable." << endl;

if (temp->left->nullable)
   temp->first = temp->left->first + temp->right->first;
else
   temp->first = temp->left->first;
cout << "Firstpos: {";
for (int j = 0; j < temp->first.size(); j++)
{
   cout << temp->first[j] << ", ";
}
cout << "}." << endl;

if (temp->right->nullable)
   temp->last = temp->right->last + temp->left->last;
else
   temp->last = temp->right->last;
cout << "Lastpos: {";
for (int j = 0; j < temp->last.size(); j++)
{
   cout << temp->last[j] << ", ";
}
cout << "}." << endl
```

```
        << endl;

    for (int j = 0; j < temp->left->last.size(); j++)
    {
        follow[temp->left->last[j]] = follow[temp->left->last[j]] + temp->right->first;
    }
}
else if (t == '|')
{
    cout << "This isn't a leaf node. It contains the character: '|' (or). ";
    temp = new tree(t, -1);
    temp->right = st.top();
    st.pop();
    temp->left = st.top();
    st.pop();
    temp->nullable = temp->left->nullable || temp->right->nullable;
    if (temp->nullable)
        cout << "The node is nullable." << endl;
    else
        cout << "The node isn't nullable." << endl;

    temp->first = temp->left->first + temp->right->first;
    cout << "Firstpos: {";
    for (int j = 0; j < temp->first.size(); j++)
    {
        cout << temp->first[j] << ", ";
    }
    cout << "}." << endl;
```

```
            temp->last = temp->right->last + temp->left->last;
            cout << "Lastpos: {";
            for (int j = 0; j < temp->last.size(); j++)
            {
                cout << temp->last[j] << ", ";
            }
            cout << "}." << endl
                 << endl;
        }
        st.push(temp);
    }
    cout << endl;
    for (int i = 0; i < leafno; i++)
    {
        cout << "Leaf number: " << i << ". ";
        cout << "Character: " << chars[i] << "." << endl;
        cout << "Followpos: {";
        for (int j = 0; j < follow[i].size(); j++)
            cout << follow[i][j] << ", ";
        cout << "}." << endl
             << endl;
    }
    return 0;
}
```

## Inputs:

1. (a|b)*a(a|b)(a|b)
2. (a|b)*a(a|b)*
3. (a|b)*|(ac)*

## Output:

```
Enter a Regular Expression in infix form: (a|b)*.a.(a|b).(a|b)

Postfix Expression: ab|*a.ab|.ab|.#.


Character at leaf node number 0 is: a. This node isn't nullable.

Firstpos: {0}.

Lastpos: {0}.


Character at leaf node number 1 is: b. This node isn't nullable.

Firstpos: {1}.

Lastpos: {1}.


This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.

Firstpos: {0, 1, }.

Lastpos: {0, 1, }.


This isn't a leaf node. It contains the character: '*' (star). It is nullable.

Firstpos: {0, 1, }.

Lastpos: {0, 1, }.


Character at leaf node number 2 is: a. This node isn't nullable.

Firstpos: {2}.

Lastpos: {2}.
```

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {2, }.

Character at leaf node number 3 is: a. This node isn't nullable.

Firstpos: {3}.

Lastpos: {3}.

Character at leaf node number 4 is: b. This node isn't nullable.

Firstpos: {4}.

Lastpos: {4}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.

Firstpos: {3, 4, }.

Lastpos: {3, 4, }.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {3, 4, }.

Character at leaf node number 5 is: a. This node isn't nullable.

Firstpos: {5}.

Lastpos: {5}.

Character at leaf node number 6 is: b. This node isn't nullable.

Firstpos: {6}.

Lastpos: {6}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.

Firstpos: {5, 6, }.

Lastpos: {5, 6, }.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {5, 6, }.

Character at leaf node number 7 is: #. This node isn't nullable.

Firstpos: {7}.

Lastpos: {7}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {7, }.

Leaf number: 0. Character: a.

Followpos: {0, 1, 2, }.

Leaf number: 1. Character: b.

Followpos: {0, 1, 2, }.

Leaf number: 2. Character: a.

Followpos: {3, 4, }.

Leaf number: 3. Character: a.

Followpos: {5, 6, }.

Leaf number: 4. Character: b.

Followpos: {5, 6, }.

Leaf number: 5. Character: a.

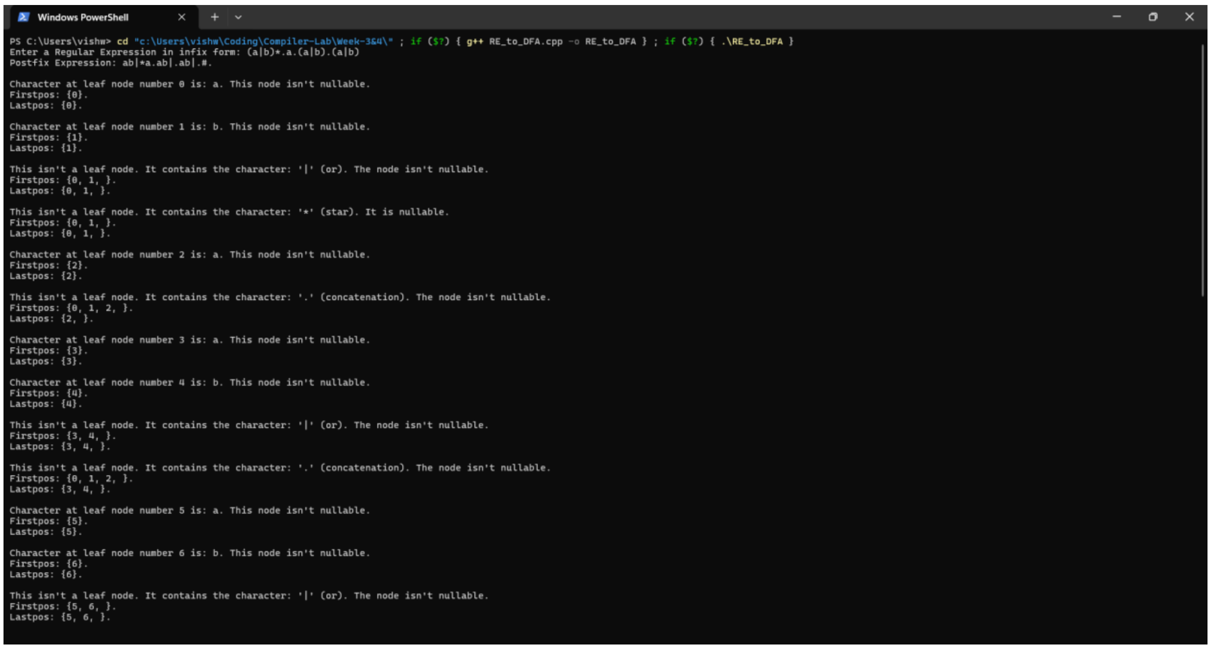
Followpos: {7, }.

Leaf number: 6. Character: b.

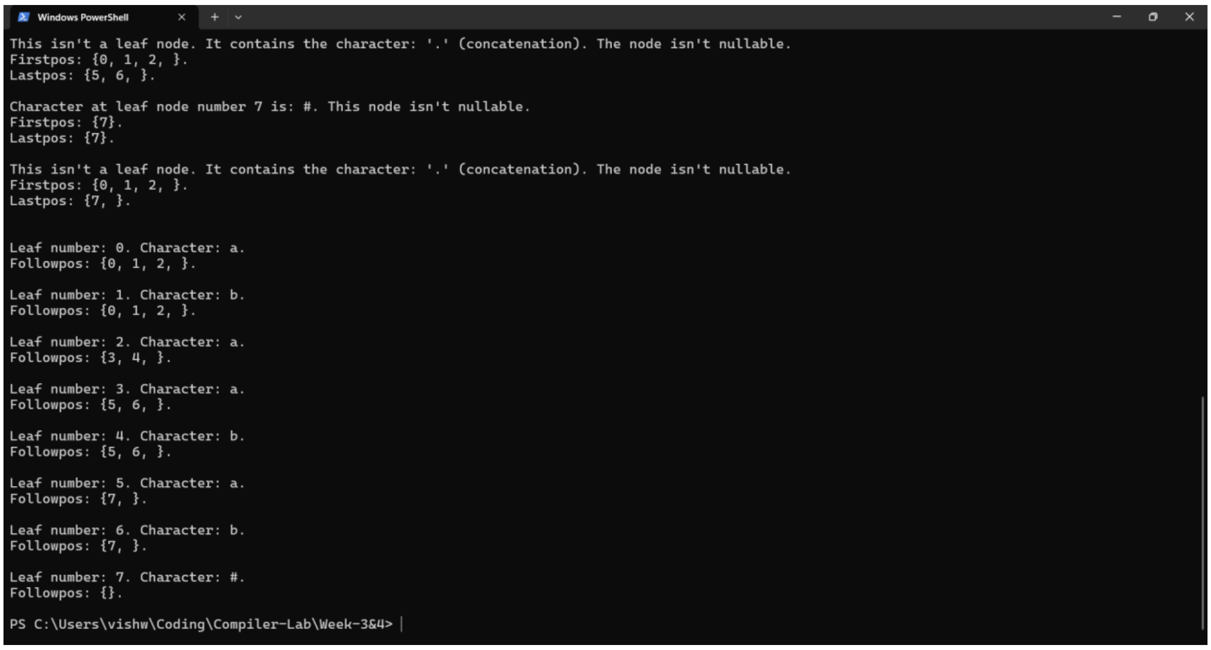
Followpos: {7, }.

Leaf number: 7. Character: #.

Followpos: {}.



```
PS C:\Users\vishw> cd "c:\Users\vishw\Coding\Compiler-Lab\Week-3&4\" ; if ($?) { g++ RE_to_DFA.cpp -o RE_to_DFA } ; if ($?) { .\RE_to_DFA }
Enter a Regular Expression in infix form: (a|b)*.a.(a|b).(a|b)
Postfix Expression: ab|*a.ab|.ab|.#.

Character at leaf node number 0 is: a. This node isn't nullable.
Firstpos: {0}.
Lastpos: {0}.

Character at leaf node number 1 is: b. This node isn't nullable.
Firstpos: {1}.
Lastpos: {1}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.
Firstpos: {0, 1, }.
Lastpos: {0, 1, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.
Firstpos: {0, 1, }.
Lastpos: {0, 1, }.

Character at leaf node number 2 is: a. This node isn't nullable.
Firstpos: {2}.
Lastpos: {2}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {2, }.

Character at leaf node number 3 is: a. This node isn't nullable.
Firstpos: {3}.
Lastpos: {3}.

Character at leaf node number 4 is: b. This node isn't nullable.
Firstpos: {4}.
Lastpos: {4}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.
Firstpos: {3, 4, }.
Lastpos: {3, 4, }.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {3, 4, }.

Character at leaf node number 5 is: a. This node isn't nullable.
Firstpos: {5}.
Lastpos: {5}.

Character at leaf node number 6 is: b. This node isn't nullable.
Firstpos: {6}.
Lastpos: {6}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.
Firstpos: {5, 6, }.
Lastpos: {5, 6, }.
```



```
This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {5, 6, }.

Character at leaf node number 7 is: #. This node isn't nullable.
Firstpos: {7}.
Lastpos: {7}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {7, }.


Leaf number: 0. Character: a.
Followpos: {0, 1, 2, }.

Leaf number: 1. Character: b.
Followpos: {0, 1, 2, }.

Leaf number: 2. Character: a.
Followpos: {3, 4, }.

Leaf number: 3. Character: a.
Followpos: {5, 6, }.

Leaf number: 4. Character: b.
Followpos: {5, 6, }.

Leaf number: 5. Character: a.
Followpos: {7, }.

Leaf number: 6. Character: b.
Followpos: {7, }.

Leaf number: 7. Character: #.
Followpos: {}.

PS C:\Users\vishw\Coding\Compiler-Lab\Week-3&4>
```

Enter a Regular Expression in infix form: (a|b)*.a.(a|b)*

Postfix Expression: ab|*a.ab|*.#.

Character at leaf node number 0 is: a. This node isn't nullable.

Firstpos: {0}.

Lastpos: {0}.

Character at leaf node number 1 is: b. This node isn't nullable.

Firstpos: {1}.

Lastpos: {1}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.

Firstpos: {0, 1, }.

Lastpos: {0, 1, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.

Firstpos: {0, 1, }.

Lastpos: {0, 1, }.

Character at leaf node number 2 is: a. This node isn't nullable.

Firstpos: {2}.

Lastpos: {2}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {2, }.

Character at leaf node number 3 is: a. This node isn't nullable.

Firstpos: {3}.

Lastpos: {3}.

Character at leaf node number 4 is: b. This node isn't nullable.

Firstpos: {4}.

Lastpos: {4}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.

Firstpos: {3, 4, }.

Lastpos: {3, 4, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.

Firstpos: {3, 4, }.

Lastpos: {3, 4, }.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {2, 3, 4, }.

Character at leaf node number 5 is: #. This node isn't nullable.

Firstpos: {5}.

Lastpos: {5}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {5, }.

Leaf number: 0. Character: a.

Followpos: {0, 1, 2, }.

Leaf number: 1. Character: b.

Followpos: {0, 1, 2, }.

Leaf number: 2. Character: a.

Followpos: {3, 4, 5, }.

Leaf number: 3. Character: a.

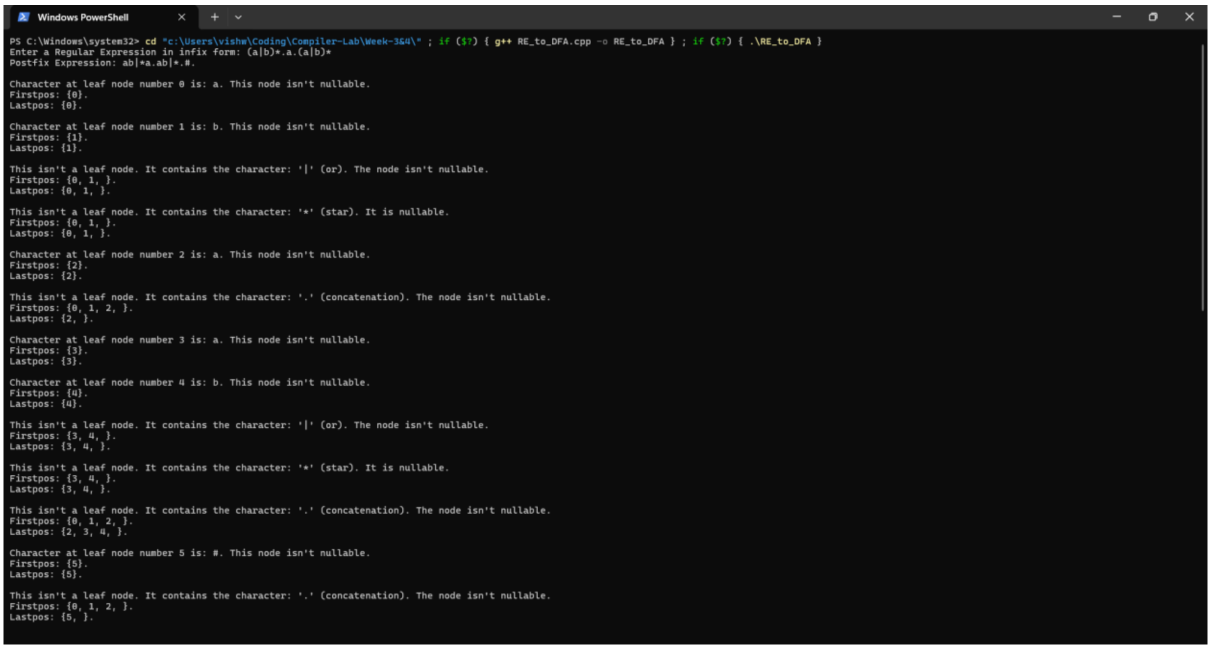
Followpos: {3, 4, 5, }.

Leaf number: 4. Character: b.

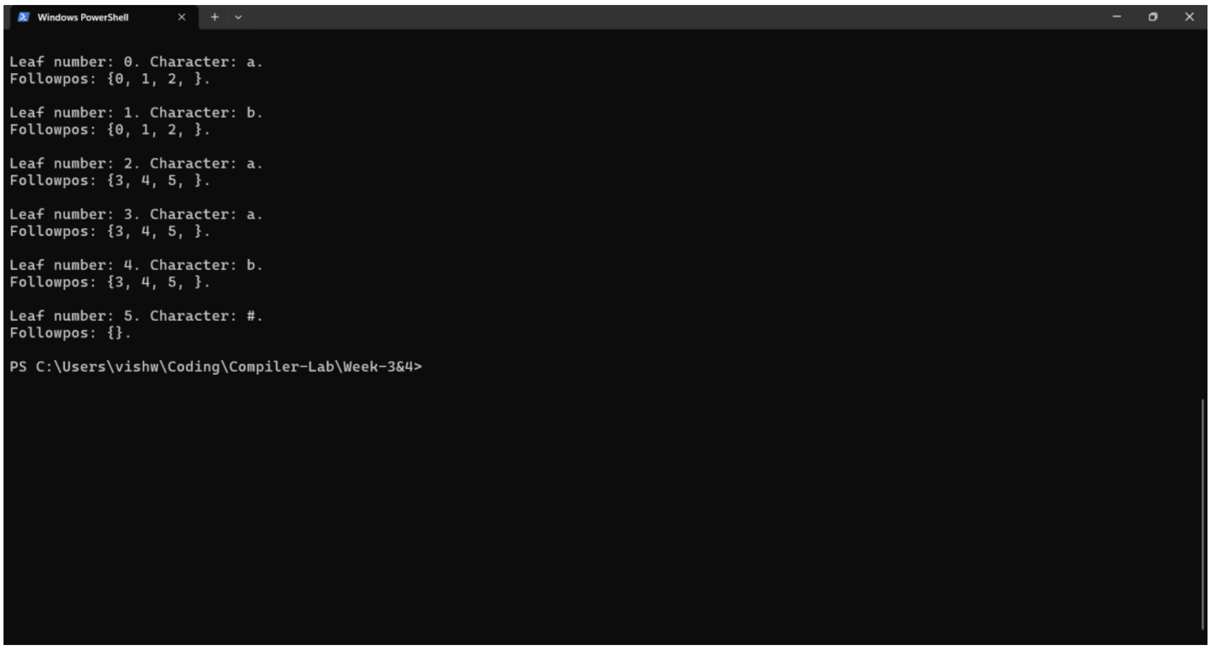
Followpos: {3, 4, 5, }.

Leaf number: 5. Character: #.

Followpos: {}.



```
PS C:\Windows\system32> cd "c:\Users\vishw\Coding\Compiler-Lab\Week-3&4\" ; if ($?) { g++ RE_to_DFA.cpp -o RE_to_DFA } ; if ($?) { .\RE_to_DFA }
Enter a Regular Expression in infix form: (a|b)*.a.(a|b)*
Postfix Expression: ab|*a.ab|*.#.

Character at leaf node number 0 is: a. This node isn't nullable.
Firstpos: {0}.
Lastpos: {0}.

Character at leaf node number 1 is: b. This node isn't nullable.
Firstpos: {1}.
Lastpos: {1}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.
Firstpos: {0, 1, }.
Lastpos: {0, 1, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.
Firstpos: {0, 1, }.
Lastpos: {0, 1, }.

Character at leaf node number 2 is: a. This node isn't nullable.
Firstpos: {2}.
Lastpos: {2}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {2, }.

Character at leaf node number 3 is: a. This node isn't nullable.
Firstpos: {3}.
Lastpos: {3}.

Character at leaf node number 4 is: b. This node isn't nullable.
Firstpos: {4}.
Lastpos: {4}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.
Firstpos: {3, 4, }.
Lastpos: {3, 4, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.
Firstpos: {3, 4, }.
Lastpos: {3, 4, }.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {2, 3, 4, }.

Character at leaf node number 5 is: #. This node isn't nullable.
Firstpos: {5}.
Lastpos: {5}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {5, }.
```



```
Leaf number: 0. Character: a.
Followpos: {0, 1, 2, }.

Leaf number: 1. Character: b.
Followpos: {0, 1, 2, }.

Leaf number: 2. Character: a.
Followpos: {3, 4, 5, }.

Leaf number: 3. Character: a.
Followpos: {3, 4, 5, }.

Leaf number: 4. Character: b.
Followpos: {3, 4, 5, }.

Leaf number: 5. Character: #.
Followpos: {}.

PS C:\Users\vishw\Coding\Compiler-Lab\Week-3&4>
```

```
Enter a Regular Expression in infix form: (a|b)*|(a.c)*

Postfix Expression: ab|*ac.*|#.


Character at leaf node number 0 is: a. This node isn't nullable.

Firstpos: {0}.

Lastpos: {0}.


Character at leaf node number 1 is: b. This node isn't nullable.
```

Firstpos: {1}.

Lastpos: {1}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.

Firstpos: {0, 1, }.

Lastpos: {0, 1, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.

Firstpos: {0, 1, }.

Lastpos: {0, 1, }.

Character at leaf node number 2 is: a. This node isn't nullable.

Firstpos: {2}.

Lastpos: {2}.

Character at leaf node number 3 is: c. This node isn't nullable.

Firstpos: {3}.

Lastpos: {3}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {2, }.

Lastpos: {3, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.

Firstpos: {2, }.

Lastpos: {3, }.

This isn't a leaf node. It contains the character: '|' (or). The node is nullable.

Firstpos: {0, 1, 2, }.

Lastpos: {0, 1, 3, }.

Character at leaf node number 4 is: #. This node isn't nullable.

Firstpos: {4}.

Lastpos: {4}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.

Firstpos: {0, 1, 2, 4, }.

Lastpos: {4, }.

Leaf number: 0. Character: a.

Followpos: {0, 1, 4, }.

Leaf number: 1. Character: b.

Followpos: {0, 1, 4, }.

Leaf number: 2. Character: a.

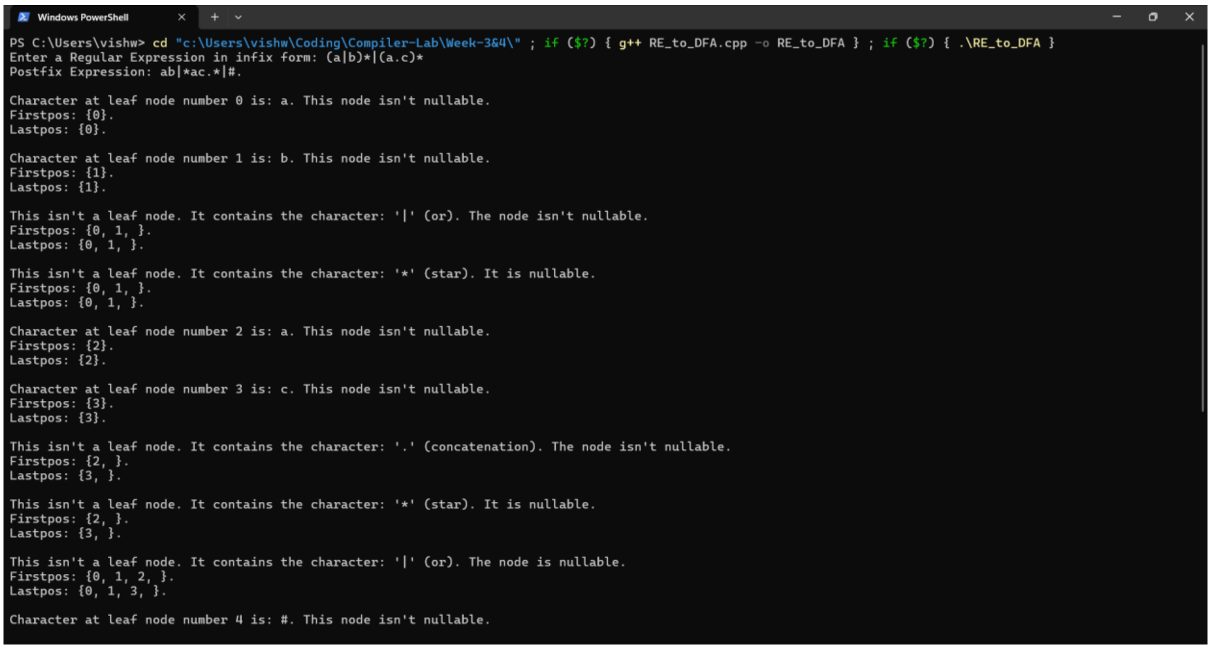
Followpos: {3, }.

Leaf number: 3. Character: c.

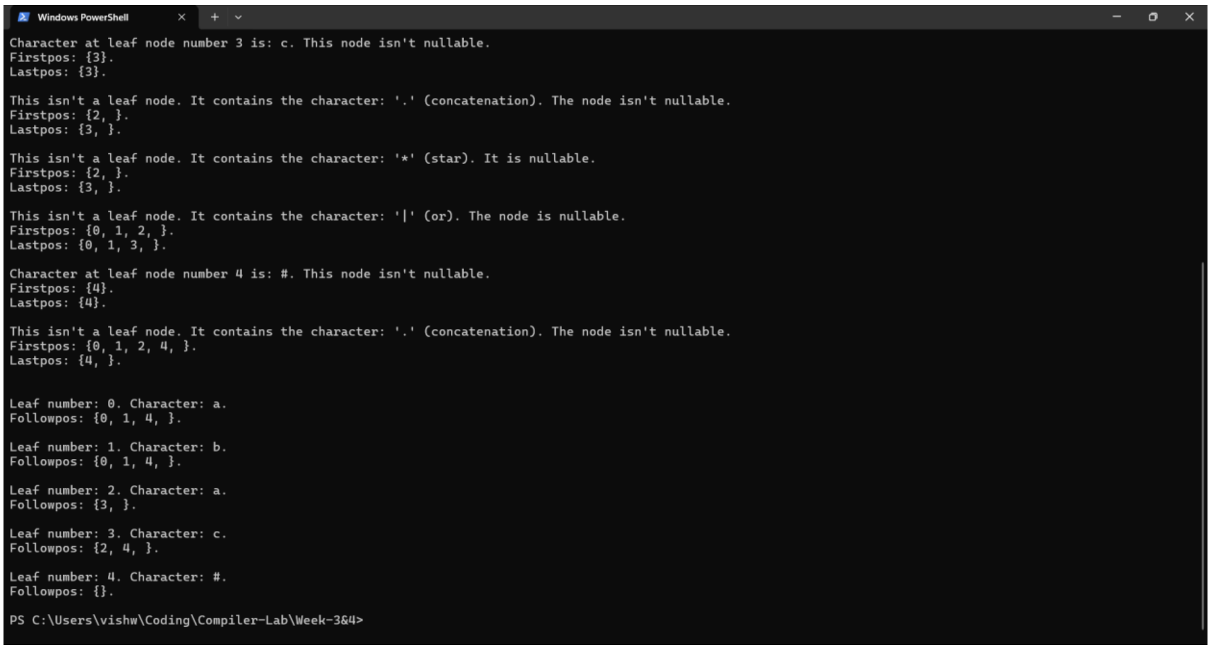
Followpos: {2, 4, }.

Leaf number: 4. Character: #.

Followpos: {}.



```
PS C:\Users\vishw> cd "c:\Users\vishw\Coding\Compiler-Lab\Week-3&4\" ; if ($?) { g++ RE_to_DFA.cpp -o RE_to_DFA } ; if ($?) { .\RE_to_DFA }
Enter a Regular Expression in infix form: (a|b)*|(a.c)*
Postfix Expression: ab|*ac.*|#.

Character at leaf node number 0 is: a. This node isn't nullable.
Firstpos: {0}.
Lastpos: {0}.

Character at leaf node number 1 is: b. This node isn't nullable.
Firstpos: {1}.
Lastpos: {1}.

This isn't a leaf node. It contains the character: '|' (or). The node isn't nullable.
Firstpos: {0, 1, }.
Lastpos: {0, 1, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.
Firstpos: {0, 1, }.
Lastpos: {0, 1, }.

Character at leaf node number 2 is: a. This node isn't nullable.
Firstpos: {2}.
Lastpos: {2}.

Character at leaf node number 3 is: c. This node isn't nullable.
Firstpos: {3}.
Lastpos: {3}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {2, }.
Lastpos: {3, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.
Firstpos: {2, }.
Lastpos: {3, }.

This isn't a leaf node. It contains the character: '|' (or). The node is nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {0, 1, 3, }.

Character at leaf node number 4 is: #. This node isn't nullable.
```



```
Character at leaf node number 3 is: c. This node isn't nullable.
Firstpos: {3}.
Lastpos: {3}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {2, }.
Lastpos: {3, }.

This isn't a leaf node. It contains the character: '*' (star). It is nullable.
Firstpos: {2, }.
Lastpos: {3, }.

This isn't a leaf node. It contains the character: '|' (or). The node is nullable.
Firstpos: {0, 1, 2, }.
Lastpos: {0, 1, 3, }.

Character at leaf node number 4 is: #. This node isn't nullable.
Firstpos: {4}.
Lastpos: {4}.

This isn't a leaf node. It contains the character: '.' (concatenation). The node isn't nullable.
Firstpos: {0, 1, 2, 4, }.
Lastpos: {4, }.


Leaf number: 0. Character: a.
Followpos: {0, 1, 4, }.

Leaf number: 1. Character: b.
Followpos: {0, 1, 4, }.

Leaf number: 2. Character: a.
Followpos: {3, }.

Leaf number: 3. Character: c.
Followpos: {2, 4, }.

Leaf number: 4. Character: #.
Followpos: {}.

PS C:\Users\vishw\Coding\Compiler-Lab\Week-3&4>
```

## Result:

The C++ program to find Nullable(), Firstpos(), Lastpos(), and Followpos() for the given Regular Expressions was successfully run and the results were verified.