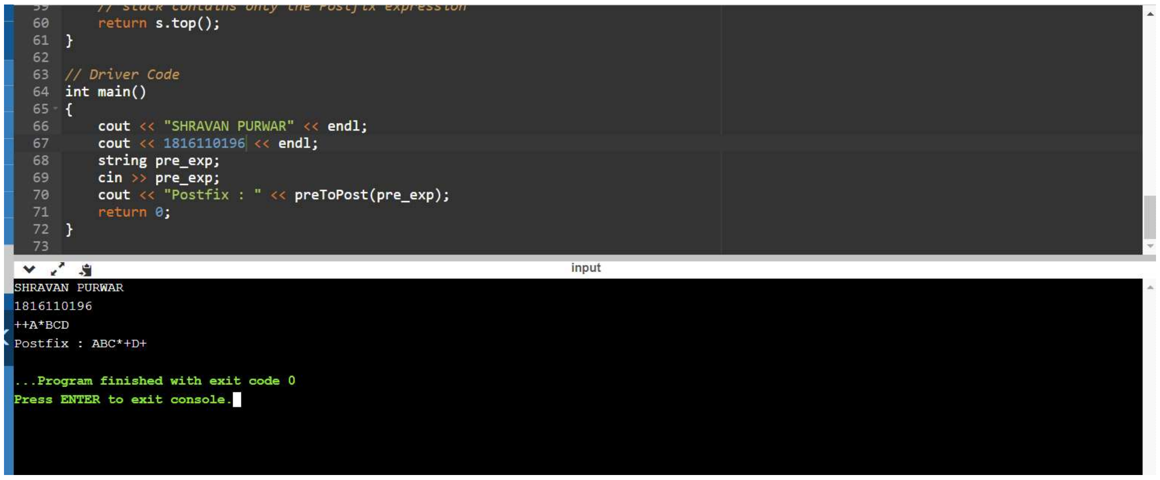
```
#include <iostream>

#include <stack>

#include <string>

using namespace std;


// funtion to check if character is operator or not

bool isOperator(char x)

{

    switch (x)

    {

    case '+':

    case '-':

    case '/':

    case '*':

        return true;

    }

    return false;

}


// Convert prefix to Postfix expression

string preToPost(string pre_exp)

{


    stack<string> s;


    // length of expression
```

```
int length = pre_exp.size();

// reading from right to left
for (int i = length - 1; i >= 0; i--)
{

   // check if symbol is operator
   if (isOperator(pre_exp[i]))
   {

      // pop two operands from stack
      string op1 = s.top();
      s.pop();
      string op2 = s.top();
      s.pop();

      // concat the operands and operator
      string temp = op1 + op2 + pre_exp[i];

      // Push string temp back to stack
      s.push(temp);
   }

   // if symbol is an operand
   else
   {
```

```
            // push the operand to the stack
            s.push(string(1, pre_exp[i]));
        }
    }

    // stack contains only the Postfix expression
    return s.top();
}

// Driver Code
int main()
{
    cout << "SHRAVAN PURWAR" << endl;
    cout << 1816110196 << endl;
    string pre_exp;
    cin >> pre_exp;
    cout << "Postfix : " << preToPost(pre_exp);
    return 0;
}
```



```cpp
    // stack contains only the Postfix expression
    return s.top();
}

// Driver Code
int main()
{
    cout << "SHRAVAN PURWAR" << endl;
    cout << 1816110196 << endl;
    string pre_exp;
    cin >> pre_exp;
    cout << "Postfix : " << preToPost(pre_exp);
    return 0;
}
```

```
SHRAVAN PURWAR
1816110196
++A*BCD
Postfix : ABC*+D+

...Program finished with exit code 0
Press ENTER to exit console.
```