EXPERIMENT 6.

UNIFICATION PROBLEM

AIM.

Implementation of unification problem and to pass all the test cases.

ALGORITHM.

STEP 1. Initialize the substitution set to be empty.

STEP 2. Recursively unify atomic sentences

i) Substitute $\frac{t_i}{v_i}$ in the existing substitution

ii) Add $\frac{t_i}{v_i}$ to the substitution setlist.

iii) If both the expression are funct the funct name must be similar and no of argument must be same in both expression.

TEST CASES.

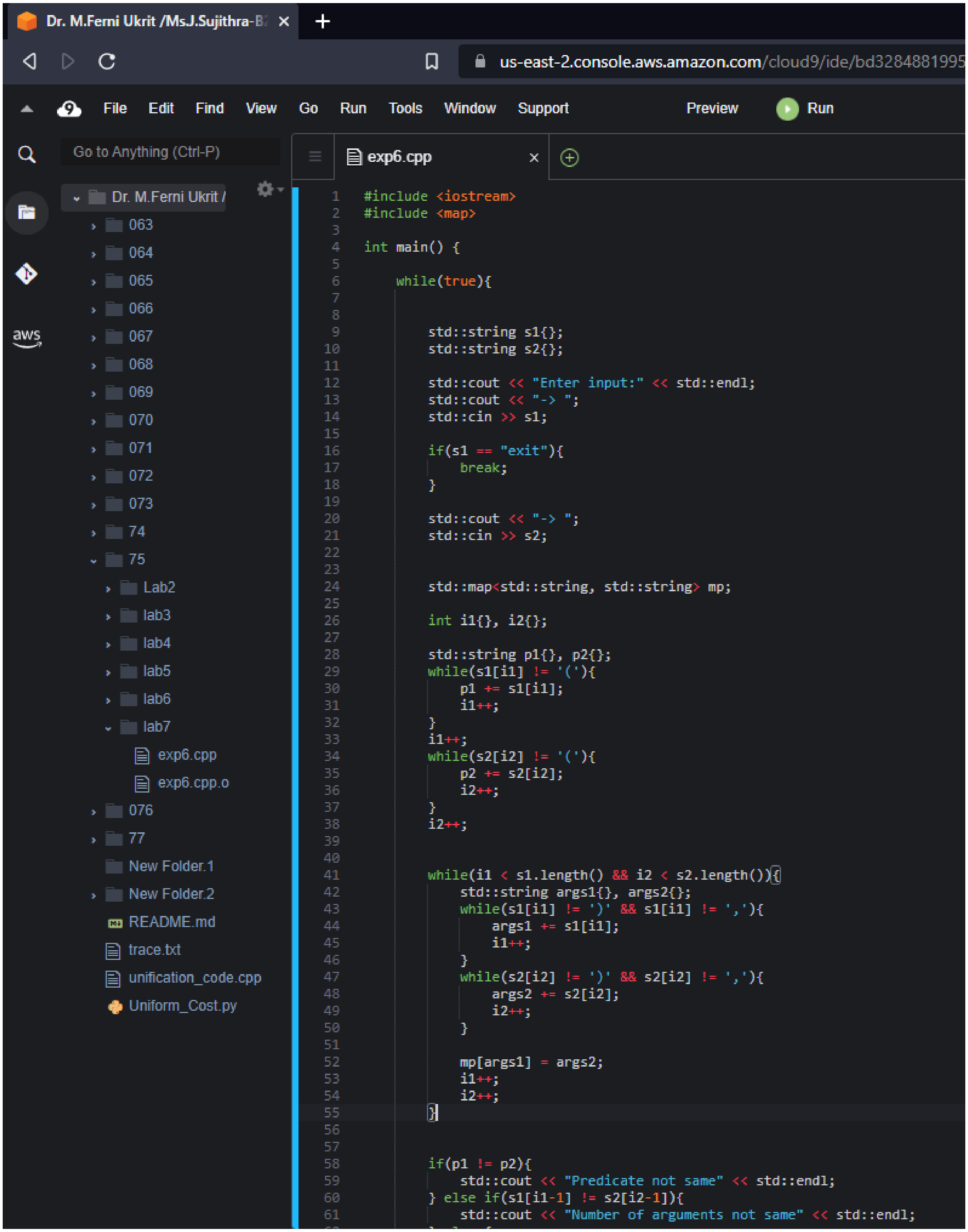
$T_{c1} = P(x,y)$ $P(t,z)$ ⇒ $\{x/t\}$, $\{\frac{y}{z}\}$

$T_{c2} = P(x,y)$ $Q(t,z)$ ⇒ predicate symbol not same

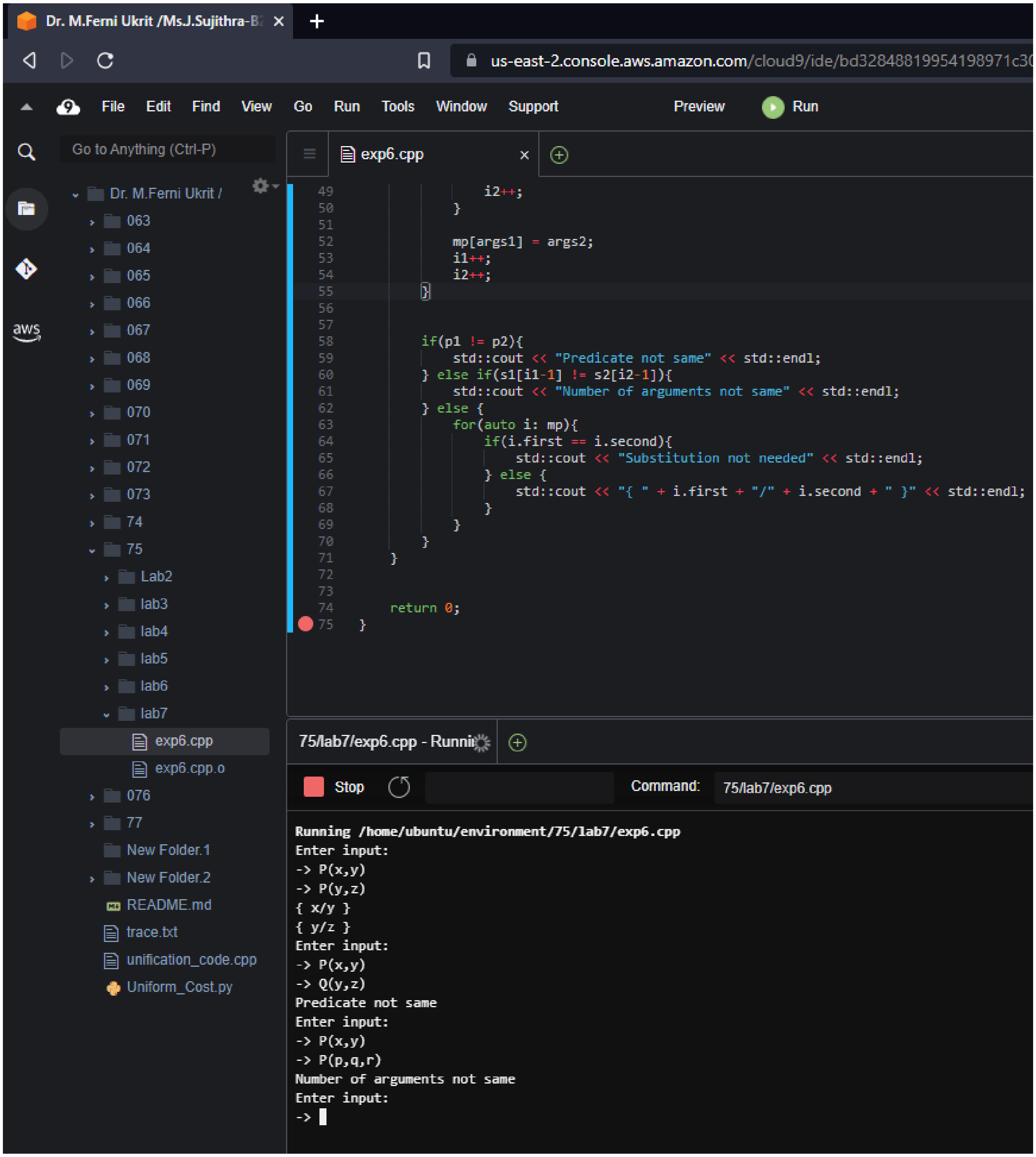
$T_{c3} = P(x,y)$ $P(t,v,z)$ ⇒ no. of argument not same

RESULT.

Implementation of unification problem is done



```cpp
#include <iostream>
#include <map>

int main() {

    while(true){


        std::string s1{};
        std::string s2{};

        std::cout << "Enter input:" << std::endl;
        std::cout << "-> ";
        std::cin >> s1;

        if(s1 == "exit"){
            break;
        }

        std::cout << "-> ";
        std::cin >> s2;


        std::map<std::string, std::string> mp;

        int i1{}, i2{};

        std::string p1{}, p2{};
        while(s1[i1] != '('){
            p1 += s1[i1];
            i1++;
        }
        i1++;
        while(s2[i2] != '('){
            p2 += s2[i2];
            i2++;
        }
        i2++;


        while(i1 < s1.length() && i2 < s2.length()){
            std::string args1{}, args2{};
            while(s1[i1] != ')' && s1[i1] != ','){
                args1 += s1[i1];
                i1++;
            }
            while(s2[i2] != ')' && s2[i2] != ','){
                args2 += s2[i2];
                i2++;
            }

            mp[args1] = args2;
            i1++;
            i2++;
        }


        if(p1 != p2){
            std::cout << "Predicate not same" << std::endl;
        } else if(s1[i1-1] != s2[i2-1]){
            std::cout << "Number of arguments not same" << std::endl;
```



```
                i2++;
            }

            mp[args1] = args2;
            i1++;
            i2++;
        }


        if(p1 != p2){
            std::cout << "Predicate not same" << std::endl;
        } else if(s1[i1-1] != s2[i2-1]){
            std::cout << "Number of arguments not same" << std::endl;
        } else {
            for(auto i: mp){
                if(i.first == i.second){
                    std::cout << "Substitution not needed" << std::endl;
                } else {
                    std::cout << "{ " + i.first + "/" + i.second + " }" << std::endl;
                }
            }
        }
    }


    return 0;
}
```

Command: 75/lab7/exp6.cpp

```
Running /home/ubuntu/environment/75/lab7/exp6.cpp
Enter input:
-> P(x,y)
-> P(y,z)
{ x/y }
{ y/z }
Enter input:
-> P(x,y)
-> Q(y,z)
Predicate not same
Enter input:
-> P(x,y)
-> P(p,q,r)
Number of arguments not same
Enter input:
->
```