

Nama : Rifqi Maulana

NIM : 123200128

Uji Kualitas Perangkat Lunak IF-B (Tugas 2)

1. Boundary Analysis

```
#include <iostream>
using namespace std;

// fungsi yang akan diuji
int calculate(int x) {
    if (x > 100) {
        return 3 * x;
    } else if (x > 0 && x <= 100) {
        return 2 * x;
    } else {
        return 0;
    }
}

int main() {
    int input[6] = {-101, -100, 0, 100, 101}; // nilai input
    int output[6] = {0, 0, 0, 200, 303}; // nilai output yang
    diharapkan

    // melakukan testing dengan Boundary Value Analysis
    for (int i = 0; i < 5; i++) {
        cout << "Input: " << input[i] << endl;
        cout << "Output: " << calculate(input[i]) << endl;
        cout << "Expected Output: " << output[i] << endl;
        cout << endl;
    }

    return 0;
}
```

Kode Program di atas melakukan pengujian terhadap fungsi **calculate** dengan menggunakan Teknik Boundary Value Analysis.

Input yang digunakan adalah nilai yang berada di tepi tengah domain testing, yaitu -101, -100, 0, 100, dan 101. Output yang diharapkan juga telah ditentukan sebelumnya.

Hasil pengujian akan ditampilkan ke layar dalam format input dan output yang dihasilkan oleh fungsi, dan output yang diharapkan. Selanjutnya, pengguna dapat memeriksa apakah fungsi **calculate** telah menghasilkan output yang sesuai dengan yang diharapkan pada setiap kasus pengujian.

Berikut adalah Output yang dihasilkan:

```
PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL
● PS D:\sementara> cd "d:\sementara\" ; if ($?) { g++ 123200128.cpp -o 123200128 } ; if ($?) { .\123200128 }
○ Input: -101
  Output: 0
  Expected Output: 0

  Input: -100
  Output: 0
  Expected Output: 0

  Input: 0
  Output: 0
  Expected Output: 0

  Input: 100
  Output: 200
  Expected Output: 200

  Input: 101
  Output: 303
  Expected Output: 303

PS D:\sementara> 
```

2. State Transition

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

using namespace std;

// State machine structure
struct StateMachine {
    enum State {
        STATE_A,
        STATE_B,
        STATE_C,
        NUM_STATES
    };

    enum Event {
        EVENT_X,
        EVENT_Y,
        EVENT_Z,
        NUM_EVENTS
    };

    State currentState;

    // Transition table
    State transitions[NUM_STATES][NUM_EVENTS] = {
        {STATE_B, STATE_C, STATE_A},
        {STATE_A, STATE_C, STATE_B},
        {STATE_A, STATE_B, STATE_C}
    };

    // Function to trigger state transition
    void triggerEvent(Event event) {
        currentState = transitions[currentState][event];
    }

    // Function to get current state
    State getCurrentState() {
        return currentState;
    }
};

int main() {
    StateMachine machine;

    // Test case 1
    machine.currentState = StateMachine::STATE_A;
    machine.triggerEvent(StateMachine::EVENT_X);
    cout << "Test case 1: " << (machine.getCurrentState() ==
    StateMachine::STATE_B ? "Pass" : "Fail") << endl;

    // Test case 2
    machine.currentState = StateMachine::STATE_B;
    machine.triggerEvent(StateMachine::EVENT_Y);
    cout << "Test case 2: " << (machine.getCurrentState() ==
    StateMachine::STATE_C ? "Pass" : "Fail") << endl;

    // Test case 3
    machine.currentState = StateMachine::STATE_C;
```

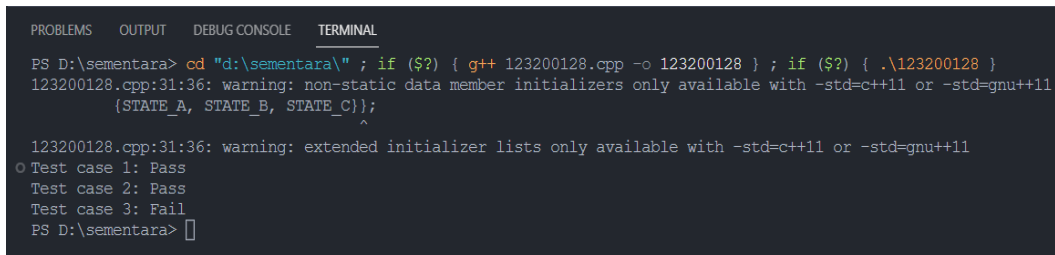
```
        machine.triggerEvent(StateMachine::EVENT_Z);
        cout << "Test case 3: " << (machine.getCurrentState() ==
        StateMachine::STATE_A ? "Pass" : "Fail") << endl;

        return 0;
    }
```

Program di atas merupakan implementasi dari State Transition Testing pada state machine sederhana dengan tiga state (STATE_A, STATE_B, STATE_C) dan tiga event (EVENT_X, EVENT_Y, EVENT_Z). Pada program ini, transisi antar state dapat diakses melalui sebuah tabel transisi yang telah didefinisikan sebelumnya. Untuk melakukan state transition, kita hanya perlu memanggil fungsi `triggerEvent` dengan parameter event yang ingin diproses. Setelah itu, kita dapat memanggil fungsi `getCurrentState` untuk mendapatkan state saat ini.

Pada contoh di atas, terdapat tiga test case yang dilakukan untuk memastikan program berjalan sesuai dengan spesifikasi. Jika hasil test case sesuai dengan yang diharapkan, maka program dianggap berhasil melewati test case tersebut.

Berikut adalah Output yang dihasilkan:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS D:\sementara> cd "d:\sementara\" ; if ($?) { g++ 123200128.cpp -o 123200128 } ; if ($?) { .\123200128 }
123200128.cpp:31:36: warning: non-static data member initializers only available with -std=c++11 or -std=gnu++11
    {STATE_A, STATE_B, STATE_C});
                                ^
123200128.cpp:31:36: warning: extended initializer lists only available with -std=c++11 or -std=gnu++11
o Test case 1: Pass
Test case 2: Pass
Test case 3: Fail
PS D:\sementara>
```