Họ và tên: Phan Mạnh Thắng

Mã sinh viên: 21020405

Đề bài:

- Bài làm cho toàn bộ các bài tập trong slide
- Báo cáo phân tích, thiết kế các ca kiểm thử, và kiểm thử chương trình của bạn với độ phủ C2
- Mã nguồn và test cases (ghi rõ link github trong báo cáo)

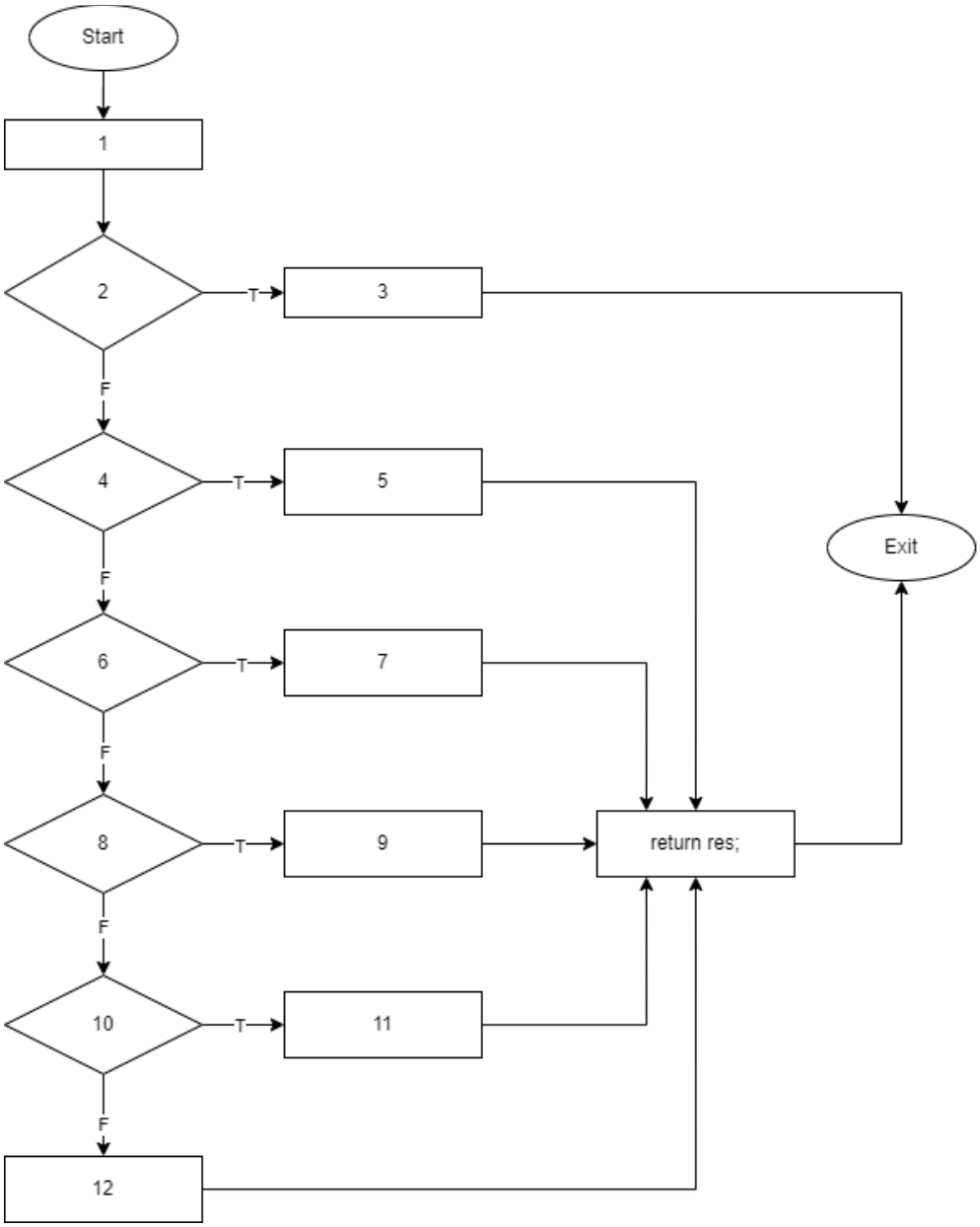
Bài 1: Trình bày các bước nhằm kiểm thử một đơn vị chương trình theo phương pháp kiểm thử dòng điều khiển với một độ đo kiểm thử cho trước.

- Tạo đồ thị dòng điều khiển tương ứng với mỗi đơn vị chương trình và mỗi đơn vị đo kiểm thử. Đối với độ đo C1 và C2, chúng tương tự nhau, tuy nhiên, khác với độ đo C3.
- Xác định các đường đi trong chương trình để đảm bảo rằng mỗi đường đi này đều thỏa mãn độ đo kiểm thử tương ứng. Mỗi đường đi sẽ tạo ra một ca kiểm thử tương ứng
- Thực hiện các ca kiểm thử đã được tạo ra trong bước 2.

Bài 2: Cho hàm được viết bằng ngôn ngữ C như trong đoạn mã dưới đây:

```
  char Grade(int score) {
1)    int res;
2)     if (score < 0 || score > 10)
3)         return 'I';
4)     if (score >= 9)
5)         return 'A';
6)     else if (score >= 8)
7)             res = 'B';
8)         else if (score >= 6.5)
9)                 res = 'C';
10)             else if (score >= 5)
11)                     res = 'D';
12)                 else res = 'F';
13)      return res;
  }
```

- Hãy xây dựng đồ thị dòng điều khiển cho hàm Grade ứng với độ đo C1 và C2



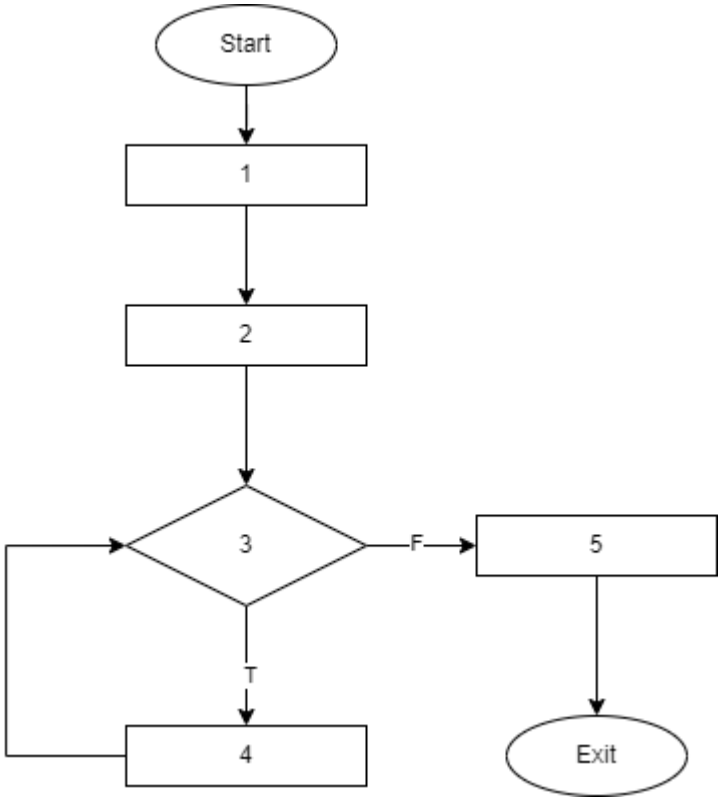
- Với độ phủ C1 (bao phủ câu lệnh), quy trình kiểm thử đòi hỏi chúng ta kiểm tra mỗi câu lệnh ít nhất một lần. Dưới đây là các trường hợp kiểm thử:
    - Test case 1: Khi score = -2, kết quả mong đợi là 'I' -> (đường đi: 0-1-
    - Test case 2: Khi score = 10, kết quả mong đợi là 'A' -> (đường đi: 0-1-3-4-12)
    - Test Case 3: Khi score = 8.5, kết quả mong đợi là 'B' -> (đường đi: 0-1-3-5-6-12)
    - Test Case 4: Khi score = 7.5, kết quả mong đợi là 'C' -> (đường đi: 0-1-3-5-7-8-12)
    - Test Case 5: Khi score = 6, kết quả mong đợi là 'D' -> (đường đi: 0-1-3-5-7-9-10-12)

- Với độ đo C2 (bao phủ nhánh), quy trình kiểm thử đòi hỏi chúng ta kiểm tra mỗi nhánh ít nhất một lần. Dưới đây là các trường hợp kiểm thử:
    - Để kiểm tra score < 0:
        - Ca kiểm thử 1: score = -1 để kiểm tra điều kiện score < 0 (đúng).
        - Ca kiểm thử 2: score = 0 để kiểm tra điều kiện score < 0 (sai).
    - Để kiểm tra socre > 10:
        - Ca kiểm thử 3: score = 11 để kiểm tra điều kiện score > 10 (đúng).
        - Ca kiểm thử 4: score = 10 để kiểm tra điều kiện score > 10 (sai).
    - Để kiểm tra socre >=9:
        - Ca kiểm thử 5: score = 9 để kiểm tra điều kiện score >= 9 (đúng).
        - Ca kiểm thử 6: score = 8 để kiểm tra điều kiện score >= 9 (sai).
    - Để kiểm tra score >=8:
        - Ca kiểm thử 7: score = 8 để kiểm tra điều kiện score >= 8 (đúng).
        - Ca kiểm thử 8: score = 7 để kiểm tra điều kiện score >= 8 (sai).
    - Để kiểm tra socre >=6:
        - Ca kiểm thử 9: score = 6.5 để kiểm tra điều kiện score >= 6.5 (đúng).
        - Ca kiểm thử 10: score = 6 để kiểm tra điều kiện score >= 6.5 (sai).
    - Để kiểm tra score >=5:
        - Ca kiểm thử 11: score = 5 để kiểm tra điều kiện score >= 5 (đúng).
        - Ca kiểm thử 12: score = 4 để kiểm tra điều kiện score >= 5 (sai).

Bài 3: Cho hàm được viết bằng ngôn ngữ C như dưới đây:

```
        int Sum (int a[], int n){
1)          int i, total = 0;
2)          for(i=0; i<n; i++)
3)              total += a[i];
4)          return total;
        }
```

- Hãy xây dựng đồ thị dòng điều khiển cho hàm Sum ứng với độ đo C1 và C2



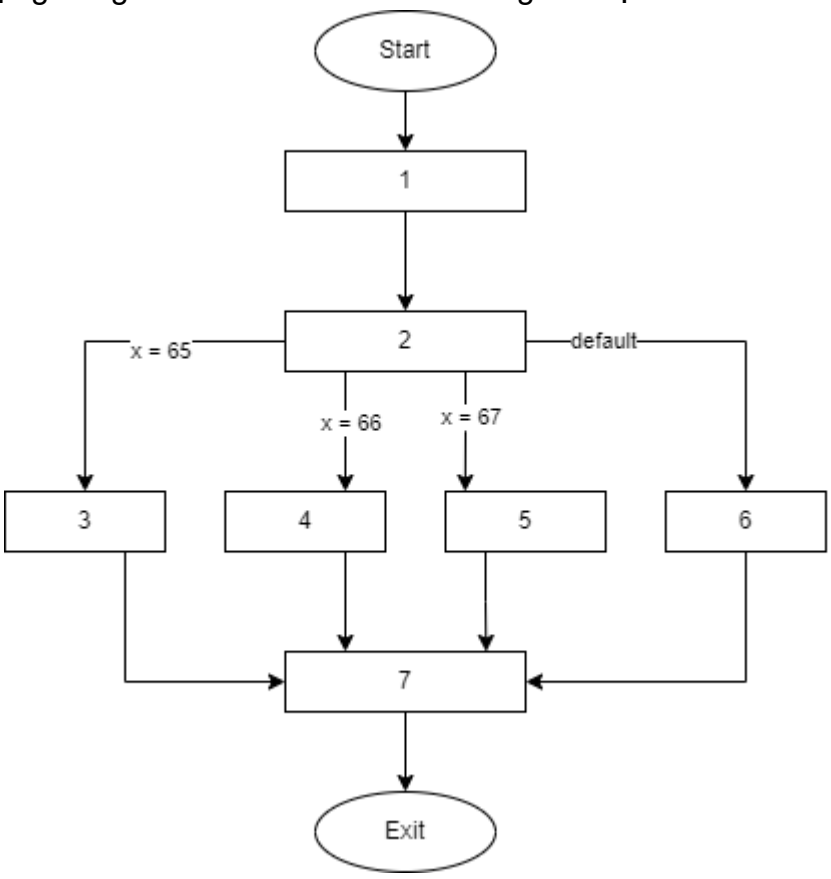
- Độ phủ C1 (bao phủ câu lệnh): Mục tiêu là đảm bảo mỗi câu lệnh trong chương trình được thực hiện ít nhất một lần.
    - Test case 1: a = [0, 1, 2], n = 3. Kết quả mong đợi là 3. Đường đi: (1-2-3-2-3-2-3-2-4)
- Độ phủ C2 (bao phủ nhánh): Mục tiêu là đảm bảo mỗi nhánh của chương trình được thực hiện ít nhất một lần.
    - Test case 2: a = [1, 2, 3], n = 3. Kết quả mong đợi là 6. Kiểm tra nhánh i < n. Đường đi: (1-2-3-2-3-2-3-2-4)
    - Test case 3: a = [], n = 0. Kết quả mong đợi là 0. Kiểm tra nhánh i >= n. Đường đi: (1-2-4)
- Kiểm thử vòng lặp for: Mục tiêu là kiểm tra các trường hợp khác nhau của vòng lặp.
    - Test case 4: a = [], n = 0. Kết quả mong đợi là 0. Kiểm tra trường hợp vòng lặp không được thực hiện. Đường đi: (1-2-4).
    - Test case 5: a = [1], n = 1. Kết quả mong đợi là 1. Kiểm tra trường hợp vòng lặp được thực hiện một lần. Đường đi: (1-2-3-2-4)
    - Test case 6: a = [1, 2, 3], n = 3. Kết quả mong đợi là 6. Kiểm tra trường hợp vòng lặp được thực hiện nhiều lần.

Bài 4:

```
      string foo(int x){
1)        string res;
2)        switch (x):
3)            case 65: res = "A"; break;
```

```
4)              case 66: res = "B"; break;
5)              case 67: res = "C"; break;
6)              default: res = "haven't check";
7)          return res;
        }
```

- Xây dựng dòng điều khiển cho hàm foo ứng với độ đo C2



- Hãy sinh đường đi và các ca kiểm thử ứng với độ đo C2

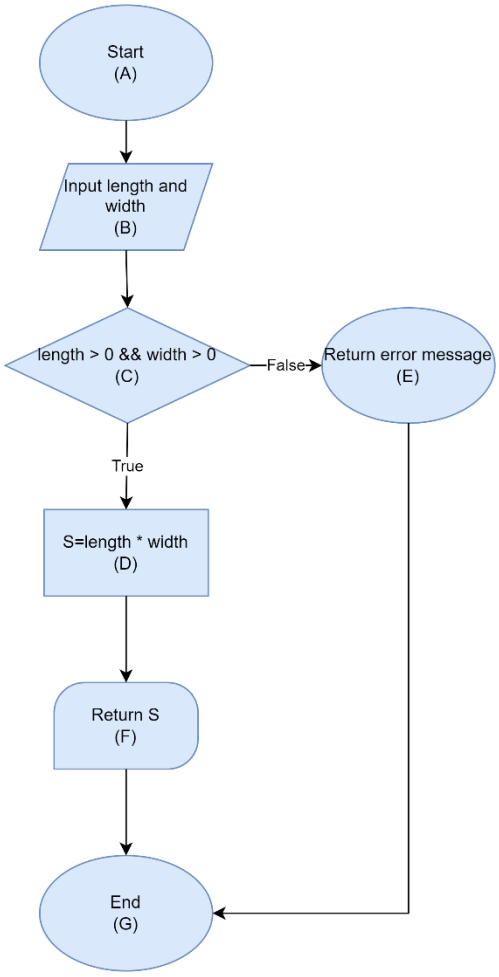
Đường đi	Input	Output	Expected output
Start - 1 - 2 - 3 - 7 - Exit	x = 65	A	A
Start - 1 - 2 - 4 - 7 - Exit	x = 66	B	B
Start - 1 - 2 - 5 - 7 - Exit	x = 67	C	C
Start - 1 - 2 - 6 - 7 - Exit	x = 64	haven’t	haven’t check

		check	

Bài 5: Kiểm thử chương trình của bạn với độ đo kiểm thử C2

Bài toán: Nhập chiều dài và chiều rộng của hình chữ nhật. Tính toán diện tích của hình đó.

- Xây dựng đồ thị điều khiển cho chương trình.
  Một đơn vị chương trình là một hàm được gọi là "area" nhận hai tham số đầu vào là chiều dài và chiều rộng của hình chữ nhật và trả về diện tích của nó.
  Đồ thị điều khiển của hàm này có thể được minh họa như sau:



- Tìm đường đi trong đồ thị thỏa mãn:
- Đối với độ bao phủ C2, cần xác định ít nhất hai đường đi: một đường đi khi điều kiện "chiều dài và chiều rộng là số dương" là đúng và một đường đi khi điều kiện là sai. Hai đường đi được chọn như sau:
  - Đường đi 1: A -> B -> C -> D -> F -> G
  - Đường đi 2: A -> B -> C -> E -> G
- Thiết lập các ca kiểm thử:

- Thiết lập các giá trị đầu vào và kết quả mong đợi cho mỗi ca kiểm thử. Dựa trên hai đường đi đã chọn, ta có thể xác định các giá trị đầu vào và kết quả mong đợi như sau:
  - Testcase 1: Với dữ liệu vào là chiều dài (length) = 10 và chiều rộng (width) = 5, kết quả mong đợi là diện tích (area) = 50.
  - Testcase 2: Với dữ liệu vào là chiều dài (length) = -1 và chiều rộng (width) = 5, kết quả mong đợi là thông báo lỗi (error message) = "Invalid input".

- **Code:**

Công nghệ:

- Ngôn ngữ: Python
- Thư viện: unittest

Chương trình tính S hình chữ nhật:

```python
def Find_S_Rectangle(width, length):
    if length <= 0 or width <= 0:
        return "Error: Length and width must be greater than 0"
    return length * width
```

Thực hiện test:

```python
import unittest

class TestArea(unittest.TestCase):
    def test_area_1(self):
        self.assertEqual(area(10, 5), 50)

    def test_area_2(self):
        self.assertEqual(area(-1, 5), "error: Invalid input")


if __name__ == '__main__':
    unittest.main()
```

Kết quả:

```
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
```

- Link mã nguồn(github): https://github.com/PhanThang0402/INT3117_2_SQT