ĐẠI HỌC CÔNG NGHỆ - ĐẠI HỌC QUỐC GIA HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP

ĐỀ BÀI: TÍNH GIÁ VÉ XEM PHIM CỦA MỖI NGƯỜI THEO ƯU ĐÃI ĐƯỢC NHẬN

Tên môn học: Kiểm thử và đảm bảo chất lượng phần mềm
Mã lớp học: INT3117_3
Giảng viên: ThS. Nguyễn Thu Trang
Sinh viên thực hiện: Nguyễn Đăng Đạo - 23021516
GitHub: software_testing_and_quality_assurance

HÀ NỘI - 2025

Mục lục

1. Giới thiệu

Hàm tinhGiaVe(int tuoi, String ngay, boolean laHocSinh, boolean laThanhVien) được sử dụng để tính giá vé xem phim dựa trên các yếu tố:

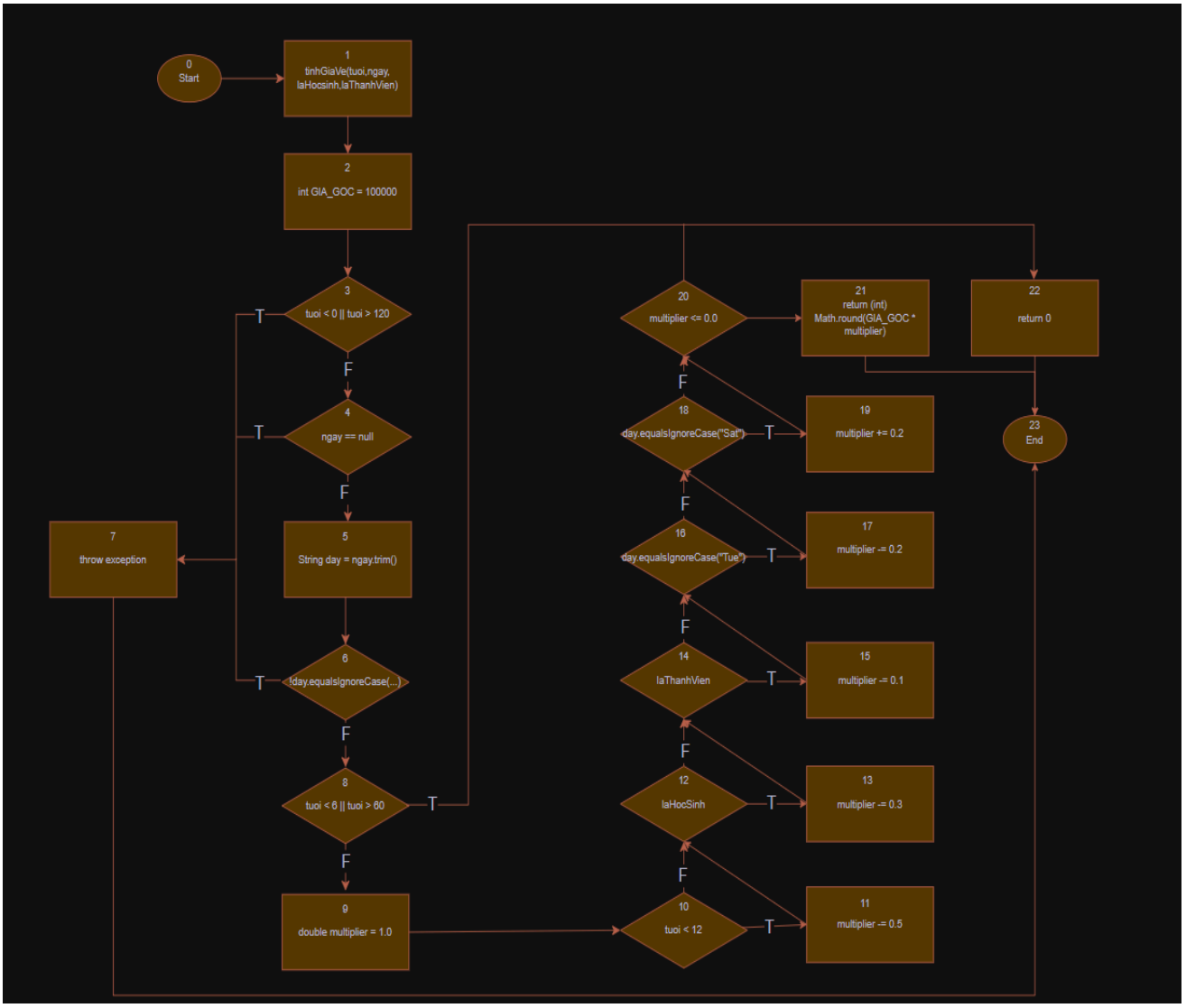
- Tuổi của khách hàng
- Ngày xem phim
- Tình trạng học sinh
- Tình trạng thành viên rạp

Mục tiêu của bài này là phân tích luồng điều khiển, xác định các nhánh logic, và thiết kế bộ ca kiểm thử đảm bảo độ bao phủ nhánh (C2) hay là mỗi điều kiện trong câu lệnh rẽ nhánh if đều được thực hiện ít nhất một lần với giá trị True và False

2. Phân tích chương trình

STT	Câu điều kiện	Ý nghĩa	Hành vi khi TRUE	Hành vi khi FALSE
1	tuoi < 0 || tuoi > 120	Tuổi không hợp lệ	Ném ngoại lệ	Tiếp tục
2	(ngay == null)	Kiểm tra ngày hợp lệ	Ném ngoại lệ	Tiếp tục
3	!(Mon/Tue/.../Sun)	Ngày không hợp lệ	Ném ngoại lệ	Tiếp tục
4	tuoi < 6 || tuoi >= 60	Miễn phí vé cho trẻ nhỏ và người già	Trả về 0	Tiếp tục
5	tuoi < 12	Giảm 50% cho trẻ nhỏ	Giảm multiplier 0.5	Tiếp tục
6	(laHocSinh)	Giảm 30% cho học sinh	Giảm multiplier 0.3	Tiếp tục
7	(laThanhVien)	Giảm 10% cho thành viên	Giảm multiplier 0.1	Tiếp tục
8	(day.equalsIgnoreCase("Tue"))	Giảm thêm 20% vào thứ Ba	Giảm multiplier 0.2	Tiếp tục
9	(day.equalsIgnoreCase("Sat"))	Tăng 20% vào thứ Bảy	Tăng multiplier 0.2	Tiếp tục
10	(multiplier <= 0.0)	Nếu giảm quá nhiều, vé miễn phí	Trả về 0	Tính giá vé bình thường

3. Đồ thị dòng điều khiển



4. Sinh đường đi với các ca kiểm thử C2

Mã đường	Đường đi (theo node trong hình)	Mô tả luồng thực thi	Input	Expected Output
P1	0 - 1 - 2 - 3(T) - 7 - 23	Tuổi < 0 => Ném ngoại lệ	(-1, "Tue", true, true)	Exception (Tuổi không hợp lệ)
P2	0 - 1 - 2 - 3(F) - 4(T) - 7 - 23	Tuổi hợp lệ, nhưng ngày null => ngoại lệ	(25, null, false, false)	Exception (Tuổi không hợp lệ)
P3	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(T) - 7 - 23	Ngày không hợp lệ => ngoại lệ	(30, "abc", false, false)	Exception (Ngày không hợp lệ: null)
P4	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(F) - 8(T) - 22 - 23	Tuổi < 6 => vé miễn phí	(5, "Mon", false, false)	Exception (Ngày không hợp lệ: Fun)
P5	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(F) - 8(F) - 9 - 10(T) - 11 - 12(F) - 14(F) - 16(F) - 18(F) - 20(F) - 21 - 23	Trẻ em <12, không là học sinh, không là thành viên => giảm 50%	(10, "Wed", false, false)	0
P6	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(F) - 8(F) - 9 - 10(F) - 12(T) - 13 - 14(F) - 16(F) - 18(F) - 20(F) - 21 - 23	Học sinh => giảm 30%	(15, "Thu", true, false)	40000
P7	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(F) - 8(F) - 9 - 10(F) - 12(F) - 14(T) - 15 - 16(F) - 18(F) - 20(F) - 21 - 23	Thành viên => giảm 10%	(25, "Fri", false, true)	70000
P8	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(F) - 8(F) - 9 - 10(F) - 12(F) - 14(F) - 16(F) - 18(T) - 17 - 20(F) - 21 - 23	Ngày thứ Ba => giảm 20%	(20, "Tue", false, false)	90000
P9	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(F) - 8(F) - 9 - 10(F) - 12(F) - 14(F) - 16(F) - 18(F) - 19 - 20(F) - 21 - 23	Ngày thứ Bảy => tăng 20%	(22, "Sat", false, false)	80000
P10	0 - 1 - 2 - 3(F) - 4(F) - 5 - 6(F) - 8(F) - 9 - 10(T) - 11 - 12(T) - 13 - 14(T) - 15 - 16(T) - 17 - 18(T) - 19 - 20(T) - 22 - 23	Tất cả ưu đãi cùng kích hoạt => multiplier ≤ 0 => vé miễn phí	(8, "Tue", true, true)	0

- Kết quả chạy test:

```
Test case 1: lỗi từ hàm tinhGiaVe: Tuổi không hợp lệ: -1
Test case 2: lỗi từ hàm tinhGiaVe: Tuổi không hợp lệ: 121
Test case 3: lỗi từ hàm tinhGiaVe: Ngày không hợp lệ:
Test case 4: lỗi từ hàm tinhGiaVe: Ngày không hợp lệ: Fun
Test case 5: tuoi=5, ngay=Wed, hocSinh=false, thanhVien=false => giaVe=0, expected=0 => PASS
Test case 6: tuoi=10, ngay=Wed, hocSinh=false, thanhVien=true => giaVe=40000, expected=40000 => PASS
Test case 7: tuoi=15, ngay=Thu, hocSinh=true, thanhVien=false => giaVe=70000, expected=70000 => PASS
Test case 8: tuoi=25, ngay=Fri, hocSinh=false, thanhVien=true => giaVe=90000, expected=90000 => PASS
Test case 9: tuoi=20, ngay=Tue, hocSinh=false, thanhVien=false => giaVe=80000, expected=80000 => PASS
Test case 10: tuoi=8, ngay=Tue, hocSinh=true, thanhVien=true => giaVe=0, expected=0 => PASS
```

- Với kết quả chạy các test trên thì chương trình đã đạt độ phủ C2

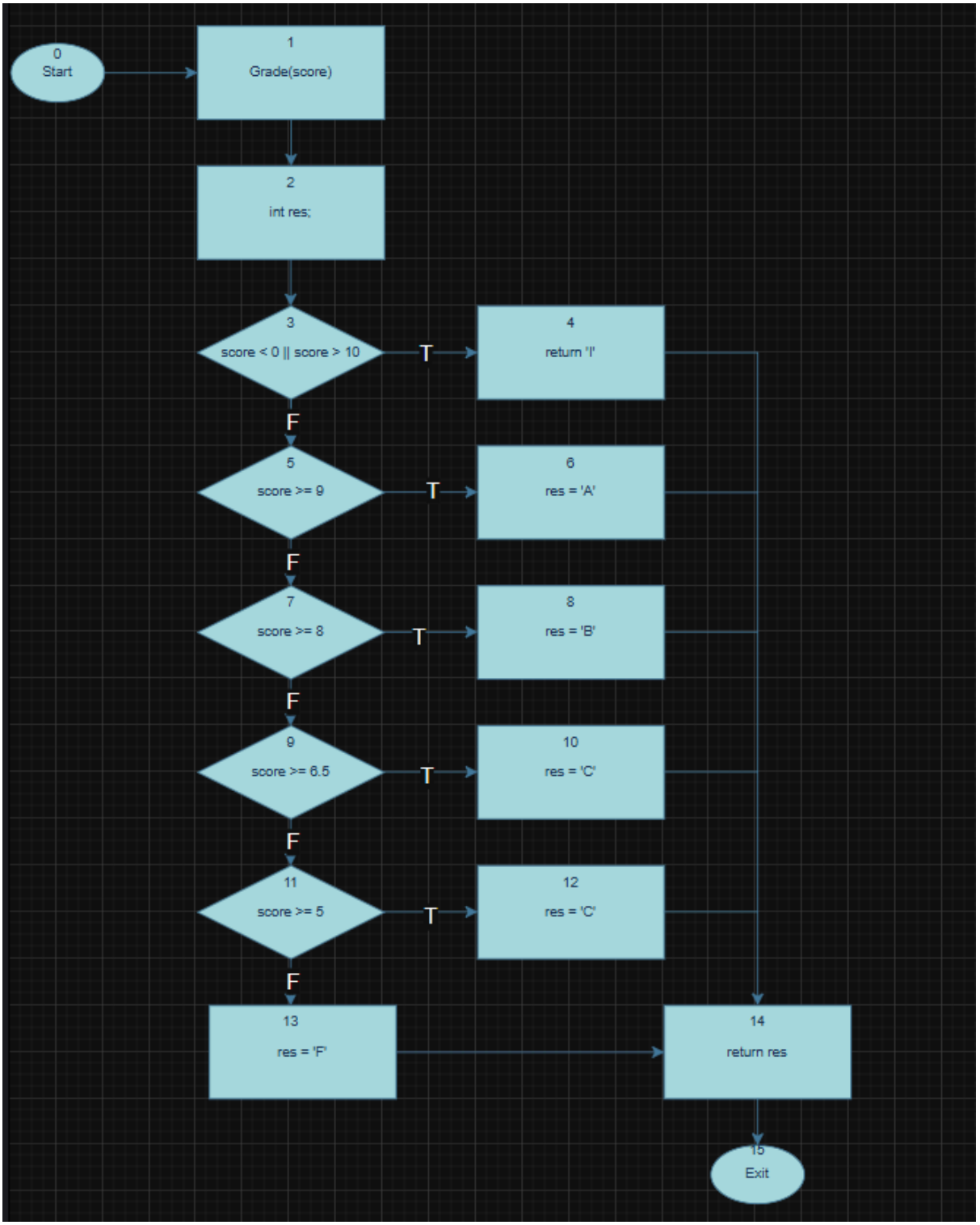
5. Các bài tập trong slide

1. Bài tập 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

   - Kiểm thử dòng điều khiển là phương pháp kiểm thử hộp trắng, dựa trên việc phân tích luồng điều khiển của chương trình để thiết kế các ca kiểm thử. Mục tiêu của phương pháp này là đảm bảo mọi luồng điều khiển quan trọng trong chương trình đều được thực thi ít nhất một lần, tùy theo độ đo kiểm thử được lựa chọn.
   - Các bước tiến hành kiểm thử:
     + Bước 1: Xây dựng đồ thị dòng điều khiển:
       - Từ mã nguồn của đơn vị chương trình, ta xác định các khối lệnh cơ bản và các luồng điều khiển giữa chúng.
       - Mỗi khối lệnh là một đỉnh trong đồ thị; các luồng điều khiển là các cạnh nối giữa các đỉnh.
     + Bước 2: Xác định độ đo kiểm thử độ đo: độ phủ C1, độ phủ C2, độ phủ C3, độ phủ câu lệnh, độ phủ vòng for,...
     + Bước 3: Xác định các đường đi cần kiểm thử: Dựa trên đồ thị dòng điều khiển, ta liệt kê các đường đi đảm bảo đạt được độ bao phủ yêu cầu.
     + Bước 4: Thiết kế các ca kiểm thử: Từ các đường đi đã xác định kết hợp đặc tả, ta chọn giá trị đầu vào phù hợp để kích hoạt từng nhánh hoặc đường đi và giá trị mong đợi.
     + Bước 5:Thực thi kiểm thử: Tiến hành chạy chương trình với từng ca kiểm thử. Ghi nhận kết quả thực tế và so sánh với kết quả mong đợi.
     + Bước 6: Đánh giá độ bao phủ: Dựa vào công cụ hoặc phân tích thủ công để xác định tỷ lệ bao phủ. Nếu độ bao phủ chưa đạt mức yêu cầu thì quay lại bước 3 để bổ sung ca kiểm thử.
     + Bước 7: Tổng kết và báo cáo: Ghi nhận các lỗi phát hiện được. Báo cáo mức độ bao phủ, chất lượng kiểm thử, và kết luận về độ tin cậy của đơn vị chương trình.

2. Bài tập 2
    - Đồ thị dòng điều khiển:



- Sinh đường đi với các ca kiểm thử C1:

  P1: 0 - 1 - 2 - 3 (T)- 4 - 14 -15

  Input: score = 11

  P2: 0 - 1 - 2- 3(F) - 5(T) - 6 - 14 - 15

  Input: score = 9.2

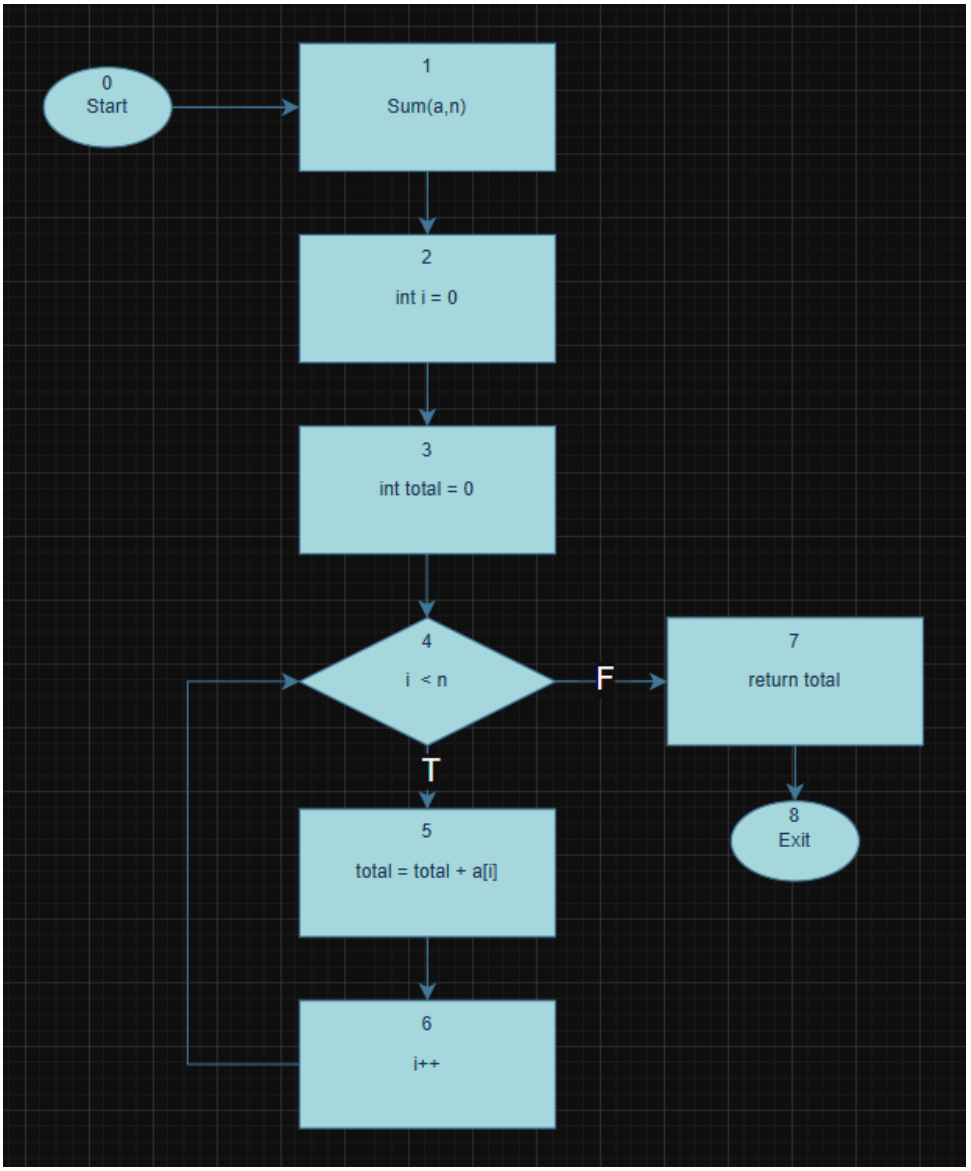
  P3: 0 - 1 - 2- 3(F) - 5(F) - 7(T) - 8 - 14 - 15

  Input: score = 8.3

P4: 0 - 1 - 2- 3(F) - 5(F) - 7(F) - 9(T) - 10 - 14 - 15
Input: score = 6.6
P5: 0 - 1 - 2- 3(F) - 5(F) - 7(F) - 9(F) - 11(T) - 12 - 14 - 15
Input: score = 5
P6: 0 - 1 - 2- 3(F) - 5(F) - 7(F) - 9(F) - 11(F) - 13 - 14 - 15
Input: score = 4.5

- Sinh đường đi với các ca kiểm thử C2: Khi thực hiện các ca kiểm thử thuộc nhóm C1, đồng thời các ca kiểm thử nhóm C2 cũng đã được bao phủ. Nguyên nhân là trong quá trình thực hiện các ca kiểm thử C1, chương trình đã đi qua toàn bộ các câu lệnh và nhánh tương ứng, do đó các đường đi kiểm thử của C2 cũng đã được thực hiện đầy đủ.

3. Bài tập 3



- Các đường đi và ca kiểm thử ứng với độ phủ C1 và C2

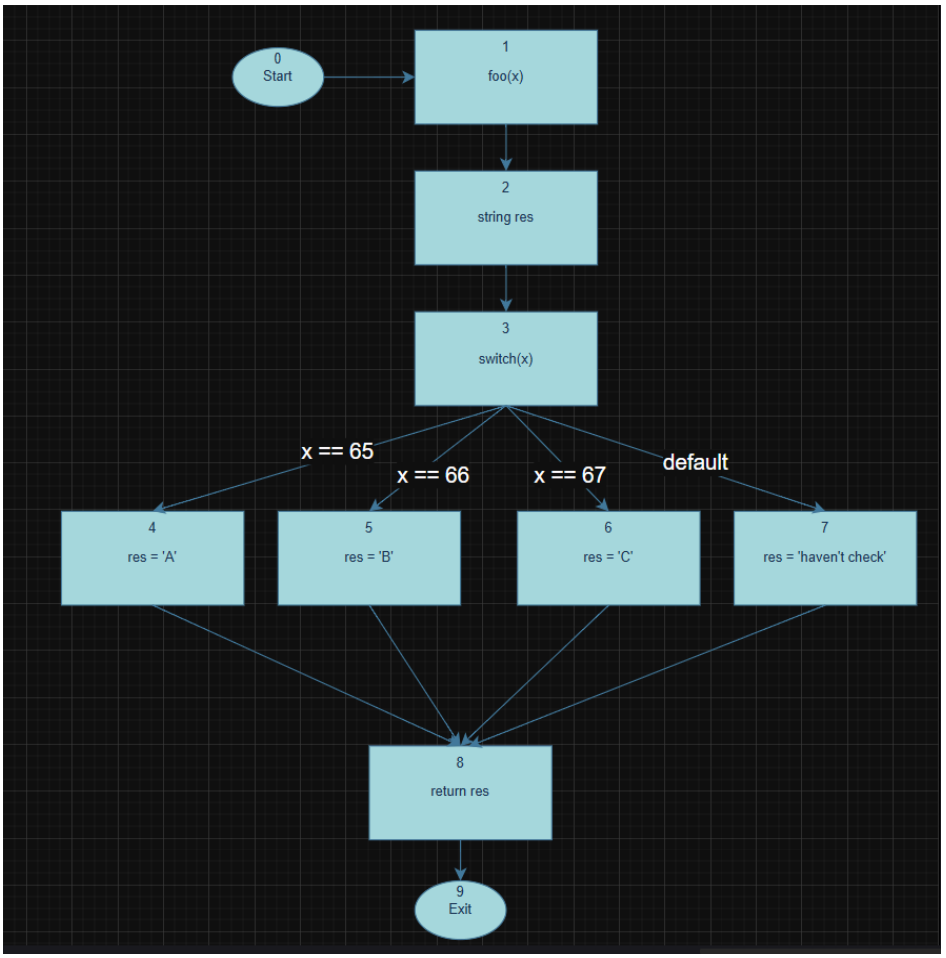
  P1: 0- 1 - 2 - 3 - 4(T) - 5 - 6 - 4(T) - 5 - 6 - 4(T) - 5 - 6 - 4(F) - 7 - 8

  Input: a[3] = {1,2,3}, n = 3

- Kiểm thử vòng lặp for: ở đây chưa xác định được số lần lặp tối đa do phụ thuộc vào biến n đầu vào, nên ta sẽ kiểm thử 4 trường hợp:
    - Vòng lặp được thực hiện 0 lần
    - Vòng lặp được thực hiện 1 lần
    - Vòng lặp được thực hiện 2 lần
    - Vòng lặp được thực hiện k lần (2 < k < số lần lặp tối đa - 1)

Số lần lặp	**Input**	**Output**
k = 0	a[] = {1, 2, 3} n = 0	0
k = 1	a[] = {1, 2, 3} n = 1	1
k = 2	a[] = {1, 2, 3} n = 2	5
k = 4	a[] = {1, 2, 3, 4, 5} n = 4	10

4. Bài tập 4



- Các đường đi và ca kiểm thử ứng với độ phủ C2:

P1: 0 - 1 - 2 - 3(65) - 4 - 8 - 9

Input: x = 65

P2: 0 - 1 - 2 - 3(66) - 5 - 8 - 9

Input: x = 66

P3: 0 - 1 - 2 - 3(67) - 6 - 8 - 9

Input: x = 67

P4: 0 - 1 - 2 - 3(89) - 7 - 8 - 9

Input: x = 89