ÔN TẬP CUỐI KỲ KIỂM THỬ

I. Quy trình kiểm thử

- Requirements/ Design Review (Nghiên cứu tài liệu mô tả, đặc tả)

- Test Planning (Kế hoạch)

- Test Designing (Thiết kế)

- Test Environment Setup (Cài đặt môi trường)

- Test Execution (Kiểm thử)

- Test Reporting (Báo cáo)

Giai đoạn	Input	Output
Requirements/ Design Review	Tài liệu *SRS, BRD, Design Doc* Thông tin trao đổi với BA, PM	Loại kiểm thử Độ ưu tiên, trọng tâm Môi trường Tính khả thi tự động *RTM*
Test Planning	*RTM* Chiến lược kế hoạch Ước tính chi phí, rủi ro, tài nguyên, môi trường Lịch trình	*Test Plan Doc* *Effort Estimation Doc* *Test Strategy* Kế hoạch, chiến lược Công cụ
Test Designing	*Test Plan Doc* *Effort Estimation Doc* *Test Strategy* *RTM*	Test Cases Test Data Test Script Cập nhật RTM
Test Environment Setup	Test Plan – Môi trường Hướng dẫn cài đặt	Test Server, Staging, DEV environment Cấu hình phần mềm Danh sách issue liên quan môi trường
Test Execution	Test Cases Test Data Môi trường kiểm thử đã sẵn sàng Build/Release cần test	Test Results (Pass/Fail/Blocked) Bug report / Defect log Bản cập nhật RTM Cập nhật tiến độ Test Execution

Test Reporting	Test results Bug reports, Defect statistics Test coverage report RTM đã cập nhật	Test Progress Report Test Summary Report Test Metrics (defect density, pass rate…) Hoàn tất chu kỳ kiểm thử Tài liệu bàn giao

SRS – Software Requirement Specification

BRD – Business Requirement Document

RTM – Requirement Traceability Matrix

Effort Estimation Doc – Tài liệu ước tính nổ lực

Test Strategy – Chiến lược

Build – Phiên bản phần mềm dùng để test

Release – Phiên bản phần mềm được đưa cho khách hàng hoặc đưa lên production

Test Metrics – Số liệu để đo chất lượng sản phẩm và hiệu quả kiểm thử

Test Progress Report	Test Summary Report
- Số lượng TC đã thiết kế, đã thực thi - Trạng thái của TCs - Số lượng lỗi phát hiện, trạng thái xử lý	- Tổng hợp kết quả của toàn bộ chu kỳ kiểm thử - Đánh giá chất lượng phần mềm với tiêu chí chấp nhận - Phân tích rủi ro còn tồn tại

II. Thiết kế Test Case – White Box Testing

White Box Testing là phương pháp kiểm tra phần mềm dựa vào thiết kế mã lệnh và cấu trúc giải thuật, dữ liệu bên trong.

Các mức độ:

- Unit: Kiểm tra đường thực thi trong một đơn vị.

- Integration: Kiểm tra đường thực thi giữa các đơn vị.

- System: Kiểm tra đường thực thi giữa các hệ thống con/ phân hệ (Subsystem)

Các kỹ thuật:

- Control Flow Testing: Kiểm thử luồng điều khiển

- Data Flow Testing: Kiểm thử luồng dữ liệu

1. Đồ thị CFG:

Các tiêu chí bao phủ

- *Statement coverage (SC)*

- *Decision coverage (DC) – Edge/Branch Coverage*

- *Condition coverage (CC)*

- Decision/condition coverage (D/CC)

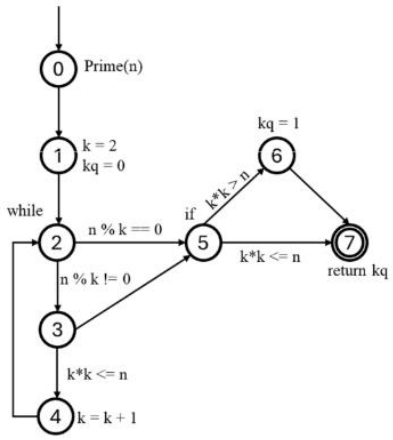
- Multiple condition coverage (MCC)

- Path coverage (PC)

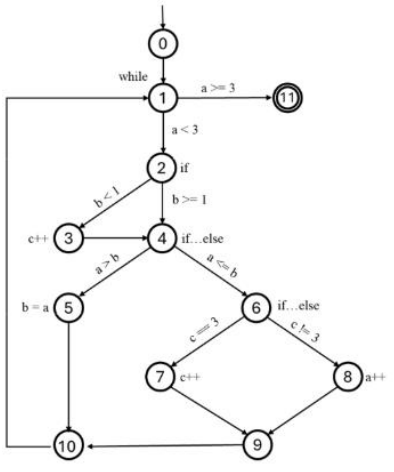
- *Basic Path coverage*

Bài tập minh hoạ

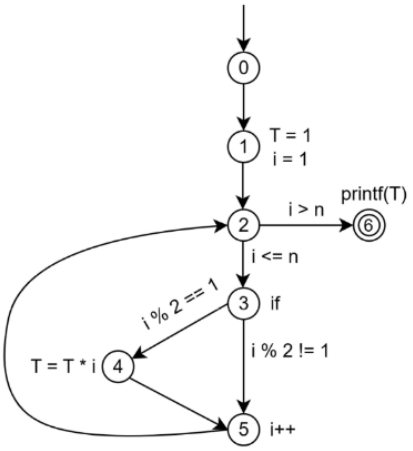
```
int Prime(long n){
    long k = 2;
    int kq = 0;
    while((n % k != 0) && (k * k <= n)){
        k = k + 1;
    }
    if(k * k > n){
        kq = 1;
    }
    return kq;
}
```



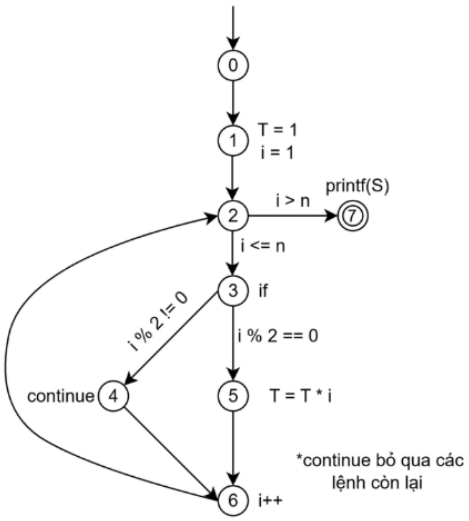

```
while(a < 3){
    if(b < 1) c++;
    if(a > b) b = a;
    else {
        if(c == 3) c++;
        else a++;
    }
}
```



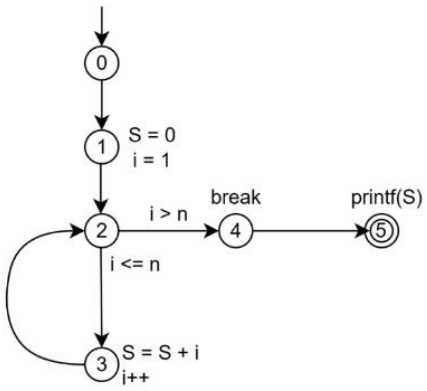
```
T = 1;
for(int i = 1; i <= n; i++){
    if(i % 2 == 1)
        T = T * i;
}
printf("T = %d", T);
```




```
T = 1;
for(int i = 1; i <= n; i++){
    if(i % 2 != 0)
        continue;
    T = T * i;
}
printf("T = %d", T);
```

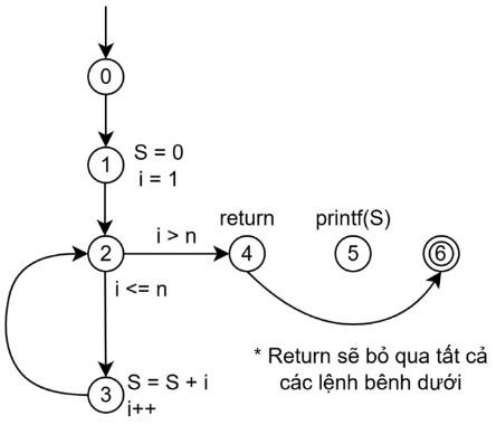



```
S = 0;
i = 1;
for(; ; i++){
    if(i > n) break;
    S = S + i;
}
printf("S = %d", S);
```

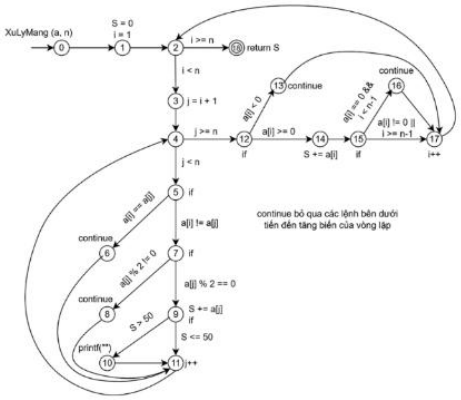


* break dừng hẳn và thoát ra

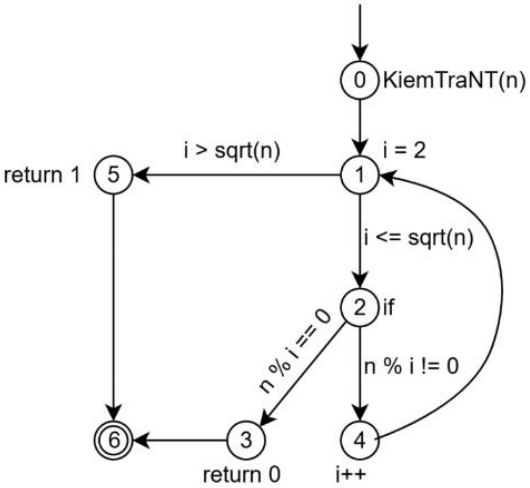
```
S = 0;
i = 1;
for(; ; i++){
    if(i > n) return;
    S = S + i;
}
printf("S = %d", S);
```



```c
int XuLyMang (int a[], int n){
    int S = 0;
    for(int i = 0; i < n; i++){
        for(int j = i + 1; j < n; j++){
            if(a[i] == a[j])
                continue;
            if(a[j] % 2 != 0)
                continue;
            S += a[j];
            if(S > 50)
                printf("Warning: Sum Exceeded 50\n");
        }
        if(a[i] < 0)
            continue
        S += a[i];
        if(a[i] == 0 && i < n - 1)
            continue;
    }
    return S;
}
```




```c
int KiemTraNT (int n){
    int i;
    for(i = 2; i <= sqrt(n); i++){
        if(n % i == 0)
            return 0;
    }
    return 1;
}
```

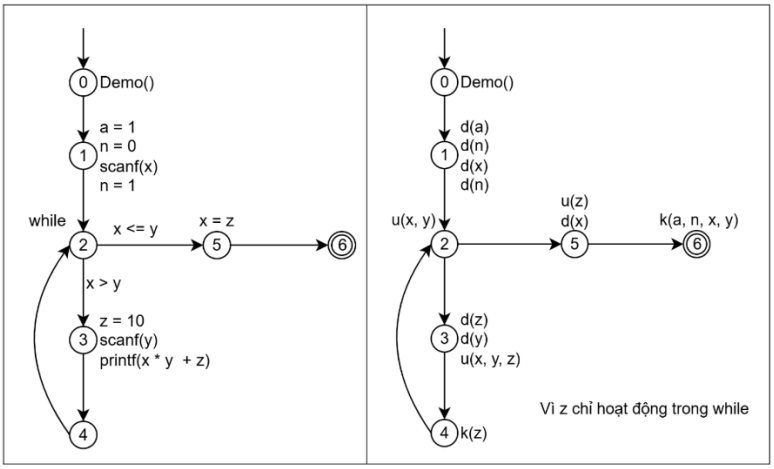


2. Đồ thị DFG

2.1 Static Data Flow Testing

Ví dụ minh hoạ

```c
void Demo(){
    int n, x, y, a = 1;
    n = 0;
    scanf("%d", &x);
    n = 1;
    while(x > y){
        int z = 10;
        scanf("%d", &y);
        printf("%d", x * y + z);
    }
    x = -z;
}
```

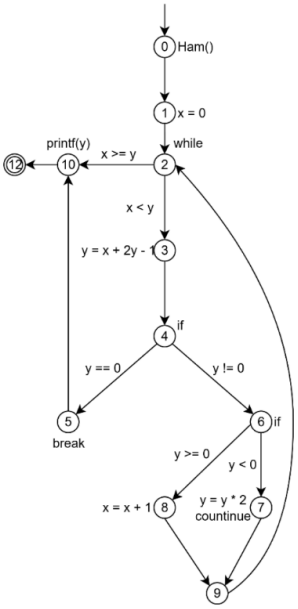


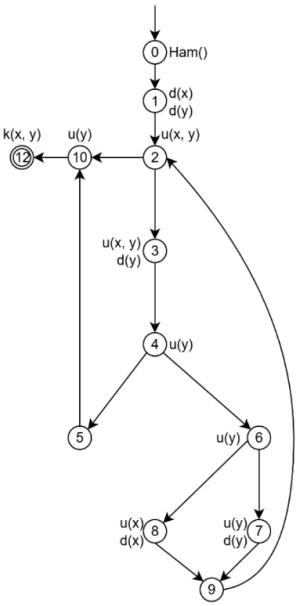
Các khả năng ghi nhận lỗi

- ~u, ~k
- d~

- dd, dk
- ku, kk

```
void Ham(){
    int y, x = 0;
    scanf("%d", &y);
    while(x < y){
        y = x + 2*y - 1;
        if(y == 0) break;
        else if(y < 0){
            y = y * 2;
            countinue;
        }
        x = x + 1;
    }
    printf("%d", y);
}
```





b. Tập lộ trình độc lập

C = 3 + 1 = 4

STT	Lộ trình	Data test
1	012(10)(11)	x = 0, y = 0 ➔ y = 0
2	01234692(10)(11)	x = 0, y = 1 ➔ y = 1
3	Không có vì điều kiện bị xung đột	
4		

c. Xây dựng kịch bản đời sống cho x, y

Biến	Lộ trình	Kịch bản
x	012(10)(11)	~duk~
	012346892(10)(11)	~duuuduk~
y	012(10)(11)	~duk~
	012346892(10)(11)	~duuduuuuk~

2.2 Dynamic Data Flow Testing

Chỉ quan tâm đến d, cu và pu

Các nút hoặc cung có u sẽ được tách ra cu hoặc pu tuỳ vào cách dùng (pu hoặc cu ở trên cung đối với Dynamic thì sẽ được ghi trên cung)

- cu: Sử dụng để tính toán, trả về, gán giá trị

Ví dụ: return a ➔ cu(a)

max = a → cu(a), d(max)

min = a + b → cu(a, b), d(min)

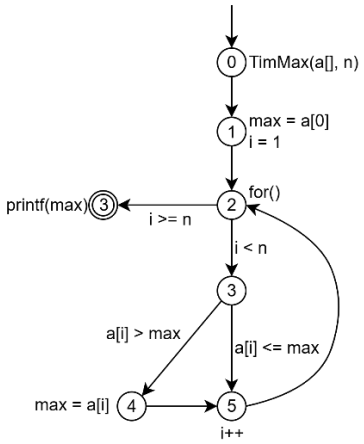
- pu: Sử dụng để so sánh

Ví dụ: a < b → pu(a, b)

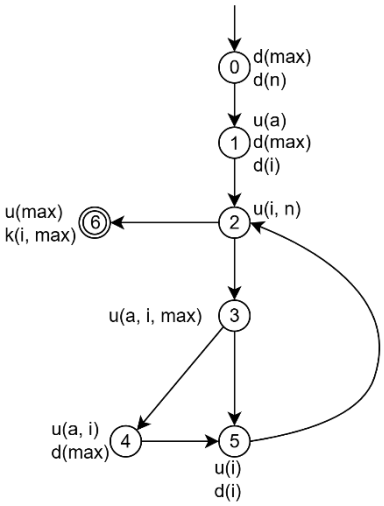
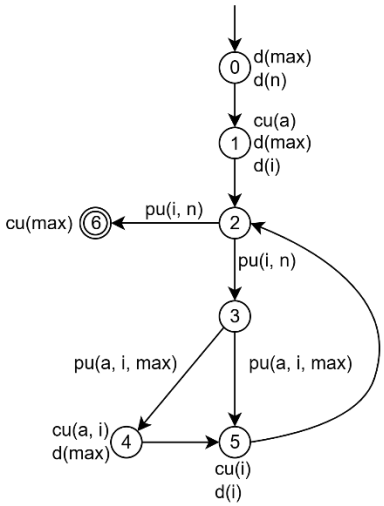
a >= max → pu(a, max)

Bài tập minh hoạ

```
void TimMax(int a[], int n){
    int max = a[0];
    int i
    for(i = 1; i<n; i++){
        if(a[i] > max){
            max = a[i];
        }
    }
    printf("%d",max);
}
```



Xây dựng đồ thị luồng dữ liệu DFG

DFG Static	DFG Dynamic
	
Tại nút 0 là d(a), d(n)	Tại nút 0 là d(a), d(n)

Tìm tất cả các du-path đối với biến max

du-pair = số d * số u

du-pair	path	du-path
(0,6)		Không thoả vì khi đi qua đỉnh 1 biến max đã def lại không thoả def clear
(0,(3,4))		
(0,(3,5))		
(1,6)	126, 123526, …	126 123526
(1,(3,4))	1234 ~~1235234~~ ….	
(1,(3,5))	1235, ~~12345,~~ …	1235
(4,6)	4526, …	4526
(4,(3,4))	~~45234,~~ …	Không thoả vì khi đi qua đỉnh 4 biến max đã def lại không thoả def clear
(4,(3,5))	~~45235~~, …	Không thoả vì chứa vòng lặp

Xây dựng dữ liệu thử thỏa mãn các tiêu chí bao phủ all-defs, all-uses, all-du-paths đối với biến max?

du-path = {126, 123526, 1234, 1235, 4526}

du(max, 1) = {126, 123526, 1234, 1235}

du(max, 4) = {4526}

Tiêu chí bao phủ All-Defs: Chọn lộ trình 0126 bao phủ được lộ trình 126 thuộc tập du(max, 1) và lộ trình 01234526 bao phủ được lộ trình 4526 của du(max, 4)

Lộ trình	Dữ liệu	Kết quả
0126	n = 1, a[] = {1}	max = 1
01234526	n = 2, a[] = {0, 1}	max = 1

du(max, 1, 6) = {126, 123526}

du(max, 1, (3, 4)) = {1234}

du(max, 1, (3, 5)) = {1235}

du(max, 4, 6) = {4526}

Tiêu chí bảo phủ thoả All-Uses:

Chọn lộ trình 0126 là lộ trình bao phủ được lô tình 126 thuộc tập du(max, 1, 6)

Chọn lộ trình 01234526 là lộ trình bao phủ được lộ trình 1234 thuộc tập du(max, 1, (3, 4)) và lộ trình 4526 thuộc tập du(max, 4, 6)

Chọn lộ trình 0123526 bao phủ được lộ trình 1235 của du(max, 1, (3, 5)) và 4526 của du(max, 4, 6)

Lộ trình	Dữ liệu	Kết quả
0126	n = 1, a[] = {1}	max = 1
01234526	n = 2 , a[] = {1, 2}	max = 2
0123526	n = 2, a[] = {1, 1}	max = 1

du-path = {126, 123526, 1235, 1234, 4526}

Tiêu chí bao phủ All-Du: Chọn lộ trình 0126, 0123526, 01234526 là các lộ trình kiểm thử bao phủ được tất cả các lô trình thuộc tập du-path.

Lộ trình	Dữ liệu	Kết quả
0126	n = 1, a[] = {1}	max = 1
0123526	n = 2, a[] = {1, 1}	max = 1
01234526	n = 2 , a[] = {1, 2}	max = 2

III. Kiểm thử tự động

Các công cụ kiểm thử tự động

- TestLink, Mantis, Bugzilla, JIRA, postman
- Selenium (IDE), Nunit, QTP
- Xenu, LoadTest, Jmeter

Khi nào nên sử dụng công cụ kiểm thử tự động?

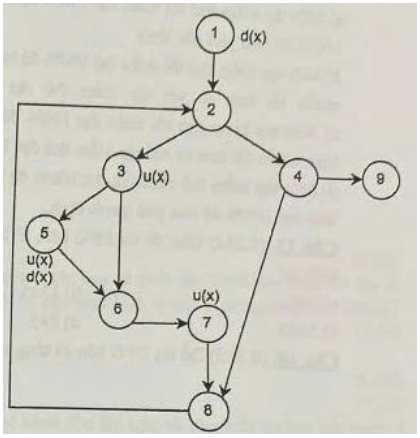
- Thực hiện kiểm thử hồi quy.
- Khi phải thực thi một số lượng test case quá lớn trong một thời gian ngắn

- Khi số lượng đầu vào cho một test case quá nhiều .
- Khi muốn thực thi các loại kiểm thử không thể kiểm tra thủ công (performance test hoặc load test,...).

TRẮC NGHIỆM

ĐỀ CLC

Câu 1. (0.25đ) Cho đồ thị luồng dữ liệu bên. Cho biết đâu là du-path đối với biến x?



a) 123, 12367, ~~5678~~

b) 1235, 5678235, 567

c) Cả a và b đều đúng

d) Cả a và b đều sai

Câu 2. (0.25đ) Liệt kê các đỉnh hội (join) trong đồ thị luồng điều bên?

~~Đỉnh 2~~

Đỉnh 6

~~Đỉnh 8~~

Câu 3. (0.25đ) Data Flow Testing tập trung vào điều gì?

a) ~~Phân tích cấu trúc dữ liệu trong hệ thống~~

b) Kiểm tra các luồng ~~điều khiển~~ trong chương trình.

c) Kiểm tra cách sử dụng và xử lý dữ liệu trong mã nguồn.

d) Tất cả các đáp án đều đúng

Câu 4. (0.25đ) Theo Tom McCabe, độ phức tạp của đồ thị luồng điều khiển G được tính theo công thức C= E-N+.... Vậy trong dấu... là gì?

a) 0

b) 1

c) 2

d) 3

Câu 5. (0.25đ) Cho đoạn mã nguồn bên:

```
void KiemTra(int a, int b, int d){
    int c;
    if(a > b)
        c = a - b;
    else
        c = a + b;

    if(c == d)
        printf("error");
}
```

Với TestCase có DataTest (a, b, d) là (5, 9, 10) thì tỉ lệ bao phủ rẽ nhánh là bao nhiêu?

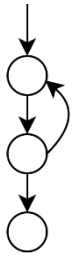
a) 50%

b) 25%

c) 75%

d) 100%

Câu 6. (0.25đ) Cho biết hình đồ thị CFG sau thể hiện cấu trúc điều khiển nào trong mã nguồn?



a) while

b) do...while

c) if...else

d) for

Câu 7. (0.25đ) Cho tập du-path đối với biến y như sau {01245, 01246, 3245, 3246, 37). Hãy cho biết tập lộ trình tối thiểu nào sau đây thỏa tiêu chí all-defs coverage?

a) {01245, 01246}

b) {3245, 37)

c) {01245, 3246}

d) {3246, 37}

Câu 8. (0.25đ) Kiểm thử hộp trắng thường được áp dụng chủ yếu vào giai đoạn nào của kiểm thử phần mềm?

a) Kiểm thử Alpha

c) Kiểm thử tích hợp

b) Kiểm thử hệ thống

d) Kiểm thử đơn vị

Câu 9. (0.25đ) Lỗi một biến được định nghĩa nhưng không được sử dụng trong bất kỳ đoạn mã nào của chương trình được phát hiện bởi cặp bất thường nào sau đây?

a) dd

b) du

c) dk

d) uu

Câu 10. (0.25đ) Công cụ kiểm thử thường được dùng để:

a) Kiểm thử chức năng

c) Kiểm thử phi chức năng

b) Kiểm thử tĩnh

d) Tất cả các đáp án đều đúng

Câu 11. (0.25đ) Cho tập lộ trình {10267, 123451, 0145347, ~~12454657~~}. Liệt kê các simple path?

Simple path: 10267, 123451, 0145347

Câu 12, (0.25đ) Tiêu chí nào sau đây không thuộc về kỹ thuật kiểm thử luồng điều khiển?

a) Bao phủ điều kiện

b) Bao phủ vòng lặp

c) Bao phủ rẽ nhánh

d) Bao phủ lộ trình

Câu 13. (0.25đ) Cho biết thông tin về việc sử dụng biến dữ liệu tại câu lệnh: c=++p-a;

a) u(p), d(p), u(a), d(c) ++p ➔ u(p) ➔ d(p)

b) u(p), u(a), d(c), u(p), d(p) a ➔ u(a)

c) u(c), u(p), u(a), d(a) c ➔ d(c)

d) u(p), d(p), u(p), u(a), d(c)

Câu 14. (0.25đ) Phát biểu nào sau đây KHÔNG đúng?

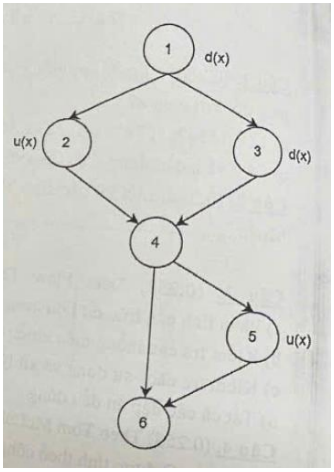
a) Một tập kiểm thử tối thiểu đạt 100% độ bao phủ lộ trình sẽ đồng thời đạt 100% độ bao phủ câu lệnh.

b) Một tập kiểm thử tối thiểu đạt 100% độ bao phủ lộ trình thường phát hiện nhiều lỗi hơn so với tập kiểm thử đạt 100% độ bao phủ câu lệnh.

c) Một tập kiểm thử tối thiểu đạt 100% độ bao phủ câu lệnh thường phát hiện nhiều lỗi hơn so với tập kiểm thử đạt 100% độ bao phủ nhánh. (TC1 yếu hơn TC2)

d) Một tập kiểm thử tối thiểu đạt 100% độ bao phủ lộ trình độc lập sẽ đồng thời đạt 100% độ bao phủ quyết định.

Câu 15. (0.25đ) Cho đồ thị DFG bên, lộ trình nào sau là def-clear đối với biến x?



a) 346 (không u(x))

b) 1345

c) 1245

d) 245

Câu 16. (0.25đ) Đồ thị DFG bên có tổng số bao nhiêu lộ trình?

4

ĐỀ TTQL & HTTT

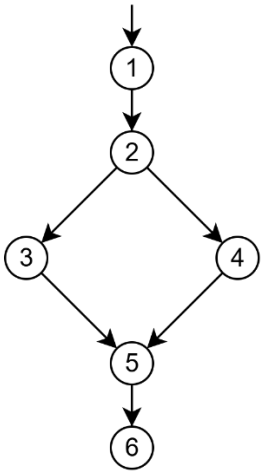
Câu 1. Cho biết đỉnh 5 (đồ thi CFG hình bên) là loại đỉnh gì trong đồ thị luồng sau đây?

a. Tuần tự

b. Quyết định

c. Kết hợp

d. Kết thúc



Câu 2. Mục tiêu chính của Control Flow Testing là gì?

a. Kiểm trạng thái của dữ liệu thông qua các xử lý

b. Đảm bảo rằng tất cả thành phần trong mã đều được kiểm tra

c. Chứng minh tính chính xác của các xử lý

d. Đo lường hiệu suất của phần mềm

Câu 3. Phương pháp BlackBox Testing còn có tên gọi khác là gì?

a. Grey Box Testing

b. Specification-Based Testing

c. Static Testing

d. Code Testing

Câu 4. Cho đoạn mã bên, cho biết số cung cần phủ để thỏa tiêu chí bao phủ quyết định?

```
printf("Ban muon tiep tuc");
scanf("%c", &d);
if(d == 'Y')
    scanf("%d%d", &a, &b);
if(a > b)
    c = a - b;
printf("%d"; c);
```

4 Cung

Câu 5. Việc xem xét (review) hoặc kiểm tra (inspection) có thể được coi là một phần của hoạt động kiểm thử không?

a. Không, vì chúng thường được áp dụng trước khi thử nghiệm

b. Không, vì chúng không thực thi phần mềm

c. Có, vì cả hai đều giúp phát hiện lỗi và cải thiện chất lượng phần mềm

d. Không, vì chúng chỉ áp dụng cho tài liệu kiểm thử

Câu 6. Trong chiến lược kiểm thứ tích hợp, ngoài cách tích hợp tăng dần (TopDown và BottomUp) thì còn chiến lược nào khác?

a. Tích hợp nhiều đơn vị (Units Test)

b. Tích hợp đồng thời (Big Bang)

c. Tích hợp theo chiều sâu (Deep)

d. Tích hợp theo chiều rộng (Width)

Câu 7. Độ phức tạp Cyclomatic của đồ thị luồng điều khiển G (C) được tính theo công thức C=P+1. Vậy P là gì của đồ thị G?

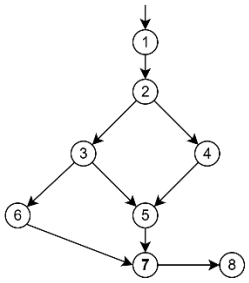
a. Số cạnh — V(G) = E – N + 2 (E số cạnh, N số nút)

b. Số đỉnh — V(G) = P + 1 (P số điểm quyết định)

c. Số đỉnh quyết định — V(G) = S + 1 (S miền khép kín)

d. Số miền khép kín của G

Câu 8. Có bao nhiêu lộ trình kiểm thử trong đồ thị luồng sau?



a. 3

b. 4

c. 5

d. 6

Câu 9. Về lý thuyết, cần thực hiện bao nhiêu TestCase để thỏa tiêu chí bao phủ đa quyết định? if (DK1 && DK2 || DK3)

a.2

b.4

c. 6

d. 8 (3 điều kiện ➔ 2^3)

Câu 10, Hãy cho biết thông tin d, u của các biến ứng câu lệnh: kq=Tong(x,y)?

a. u(x,y), d(kq) — Tong(x, y) ➔ u(x, y)

b. d(x, y, kq) kq ➔ d(kq)

c. u(x, y, kq)

d. d(kq)

Câu 11. Tại mức kiểm thử thành phần/đơn vị (Component/Unit), kỹ thuật kiểm thử phổ biến nào sau đây được áp dụng?

a. Kiểm thử bảo mật

b. Kiểm thử năng suất

c. Kiểm thử khả năng sử dụng

d. Kiểm thử bao phủ câu lệnh và nhánh

TỰ LUẬN

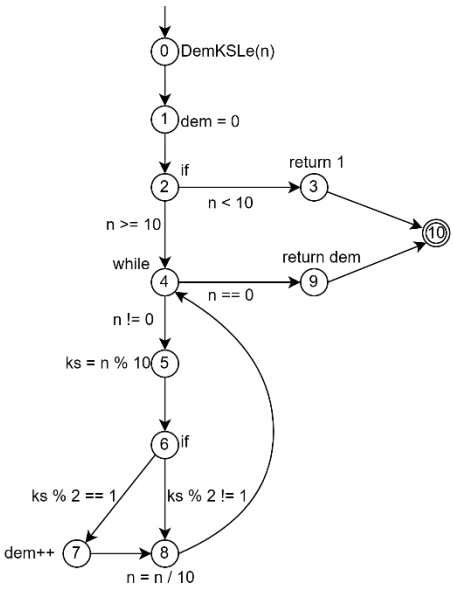
```
char DemKSLe (int n){
    char dem = 0;
    if(n < 10){
        return 1;
    }

    while(n != 0){
        char ks = n % 10;
        if(ks % 2  == 1){
            dem++;
        }
        n = n/10;
    }
    return dem;
}
```

a. Xây dựng đồ thị luồng điều khiển CFG.
b. Xác định tập lộ trình kiểm thử tối thiểu thỏa tiêu chí bao phủ quyết định, sau đó sinh các TestCase thực thi tập lộ trình này (xác định Input và Output cho TCs). Cho biết có TC nào phát hiện được lỗi trong đoạn mã không?
Chủ ý: Chọn tập lộ trình kiểm thử có thể thực thi được.
c. Xây dựng đồ thị luồng dữ liệu DFG (mức tĩnh).
d. Xác định tập lộ trình độc lập.
e. Xây dựng kịch bản kiểm thử đời sống của biến dem theo tập lộ trình độc lập trên và cho biết những bất thường trong việc sử dụng biến (nếu có) theo kỹ thuật kiểm thử luồng dữ liệu tĩnh.

Giải

a. Đồ thị luồng điều khiển CFG



b. Tập lộ trình kiểm thử tối thiểu thỏa tiêu chí bao phủ quyết định.

Paths = {0123(10), 01249(10),

01245678456849(10),

0124(56784)*9(10),

0124(5684)*9(10),…}

<u>Theo TC2:</u> lộ trình 0123(10) và 01245678456849(10) là lộ trình khả thi tối thiêu bao phủ được các nhánh của đỉnh quyết định trong đồ thị CFG.

Lộ trình 0123(10)

Input: n = 1

Output: return 1 → ER (KQ mong đợi)

return 1 → AR (KQ thực tế)

Input: n = 2

Output: return 0 → ER

return 1 → AR

➔ Với n = 2 thì phát hiện được lỗi trong đoạn mã

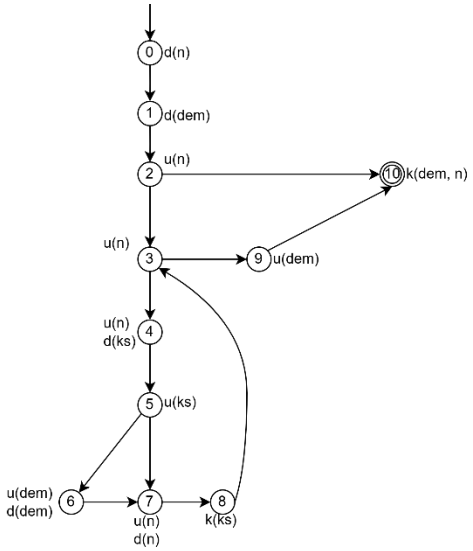
Lộ trình 01245678456849(10)

Input: n = 21

Output: dem = 1 → ER

dem = 1 → AR

c. Xây dựng đồ thị luồng dữ liệu DFG (mức tĩnh).



d. Xác định tập lộ trình độc lập.

C = 3 + 1 = 4

Lt1: 012(10)

Input: n = 1

Output: return 1

Lt2: ~~01239(10)~~

Lt3: 0123457834578 39(10)

Input: n = 20

Output: return dem = 0

Lt4: 01234567834567839(10)

Input: n = 11

Output: return dem = 2

➔ Basic paths = {012(10), 012345783457839(10), 01234567834567839(10)}

e. Xây dựng kịch bản kiểm thử đời sống của biến dem theo tập lộ trình độc lập trên và cho biết những bất thường trong việc sử dụng biến (nếu có) theo kỹ thuật kiểm thử luồng dữ liệu tĩnh.

Biến	Lộ trình	Kịch bản
dem	012(10) 012345783457839(10) 01234567834567839(10)	~dk~ (Định nghĩa nhưng chưa sử dụng) ~duk~ ~dududuk~