**BÀI TẬP VỀ KIỂM THỬ HỘP TRẮNG**

**Bài 1:** Cho hàm sau:

int linearSearch(int arr[], int target, int n)

{ **1 2 3**

for( int i = 0; i < n; i++ ) {

if ( arr[i] == target) {**(4)**

return i; **(5)**

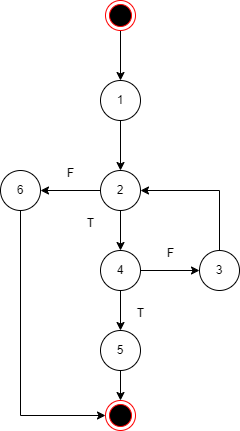
}

}

return -1; **(6)**

}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:



* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị. C= P+1=3
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.

Start -> 1 -> 2 -> 6 -> end

Start -> 1 -> 2 -> 4 -> 5 -> end

Start -> 1 -> 2 -> 4 -> 3 -> 2 -> 6 -> end

* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 2:** Cho hàm sau:

**public** **static** **void** selectionSort(**int**[] myArray) {

**int** n = myArray.length;

**for** (**int** i = 0; i < n - 1; i++) {

**int** minIndex = i;

// Find the index of the minimum element in the unsorted part of the array

**for** (**int** j = i + 1; j < n; j++) {

**if** (myArray[j] < myArray[minIndex]) {

minIndex = j;

}

}

// Swap the minimum element with the first element of the unsorted part

**int** temp = myArray[minIndex];

myArray[minIndex] = myArray[i];

myArray[i] = temp;

}

}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 3:** Cho hàm sau:

***public******static******void*** *cal(****int*** *a[],* ***int*** *n,* ***int*** *b[],* ***int*** *m,* ***int*** *c[]) {*

***int*** *k = 0, i = 0, j = 0;1*

*2 3*

***while*** *(i < n && j < m) {*

***if*** *(a[i] < b[j]) {4*

*c[k] = a[i];5*

*i++;*

*}* ***else*** *{*

*c[k] = b[j];6*

*j++;*

*}*

*k++;7*

*}*

*8*

***while*** *(i < n) {*

*c[k] = a[i];*

*i++*



*k++; 9*

*}*

*10*

***while*** *(j < m) {*

*c[k] = b[j];11*

*j++;*

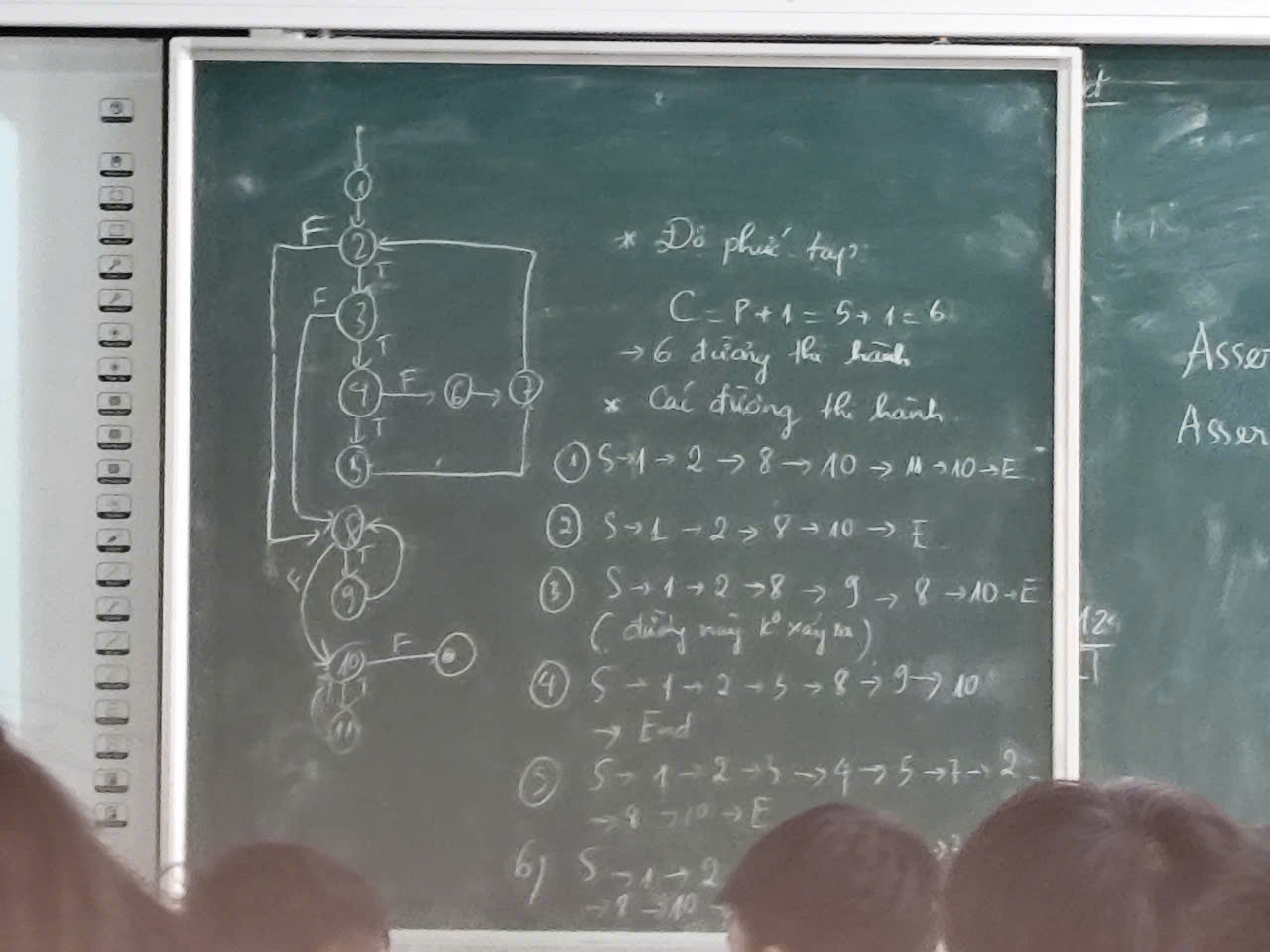
*k++;*

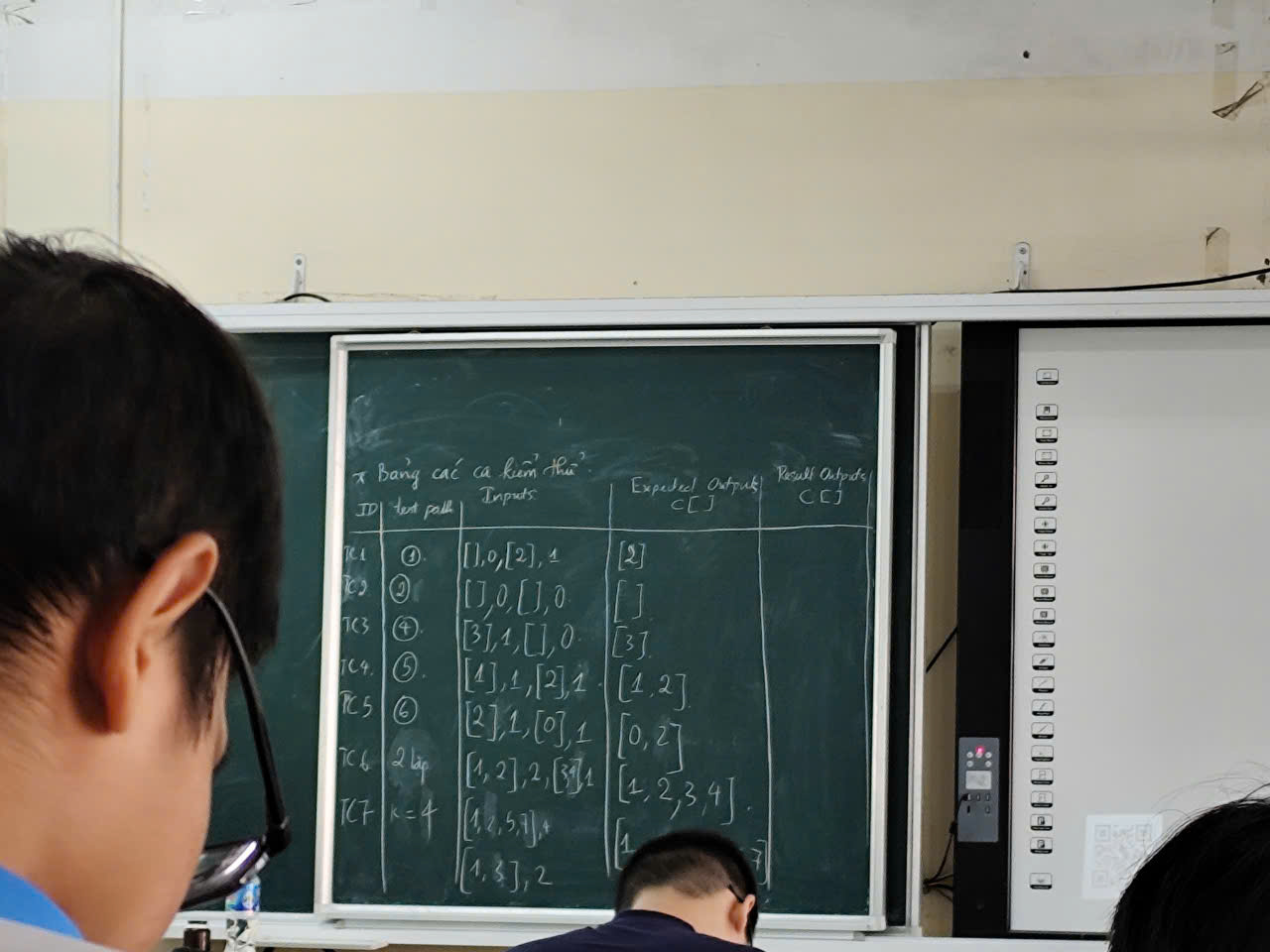
*}*

*}*

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.\





* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 4:**

Cho hàm sau:

***public******static******int*** *cal(****int*** *m1,* ***int*** *d1,* ***int*** *m2,* ***int*** *d2,* ***int*** *y) {*

***int*** *numDays;*

***if*** *(m2 == m1)*

*numDays = d2 - d1;*

***else*** *{*

***int*** *m4 = y % 4;*

***int*** *m100 = y % 100;*

***int*** *m400 = y % 400;*

***int****[] daysIn = { 0, 31, 0, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };*

***if*** *((m4 != 0) || ((m100 == 0) && (m400 != 0)))*

*daysIn[2] = 28;*

***else***

*daysIn[2] = 29;*

*numDays = d2 + (daysIn[m1] - d1);*

***int*** *i = m1 + 1;*

***while*** *(i <= m2 - 1) {*

*numDays = numDays + daysIn[i];*

*i++;*

*}*

*}*

***return*** *(numDays);*

*}*

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 5:**

**Cho hàm sau:**

int Sum(int a[], int n){

int S,i;

if(n==0)

return -1;

S=0;

i=0;

while(i<n){

S=S+a[i];

i++;

}

return S;

}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.

**Bài 6:** Cho hàm sau:

int gcd(int a, int b){

if (a == 0 || b == 0){

return a + b;

}

while (a != b){

if (a > b){

a -= b; // a = a - b

}else{

b -= a;

}

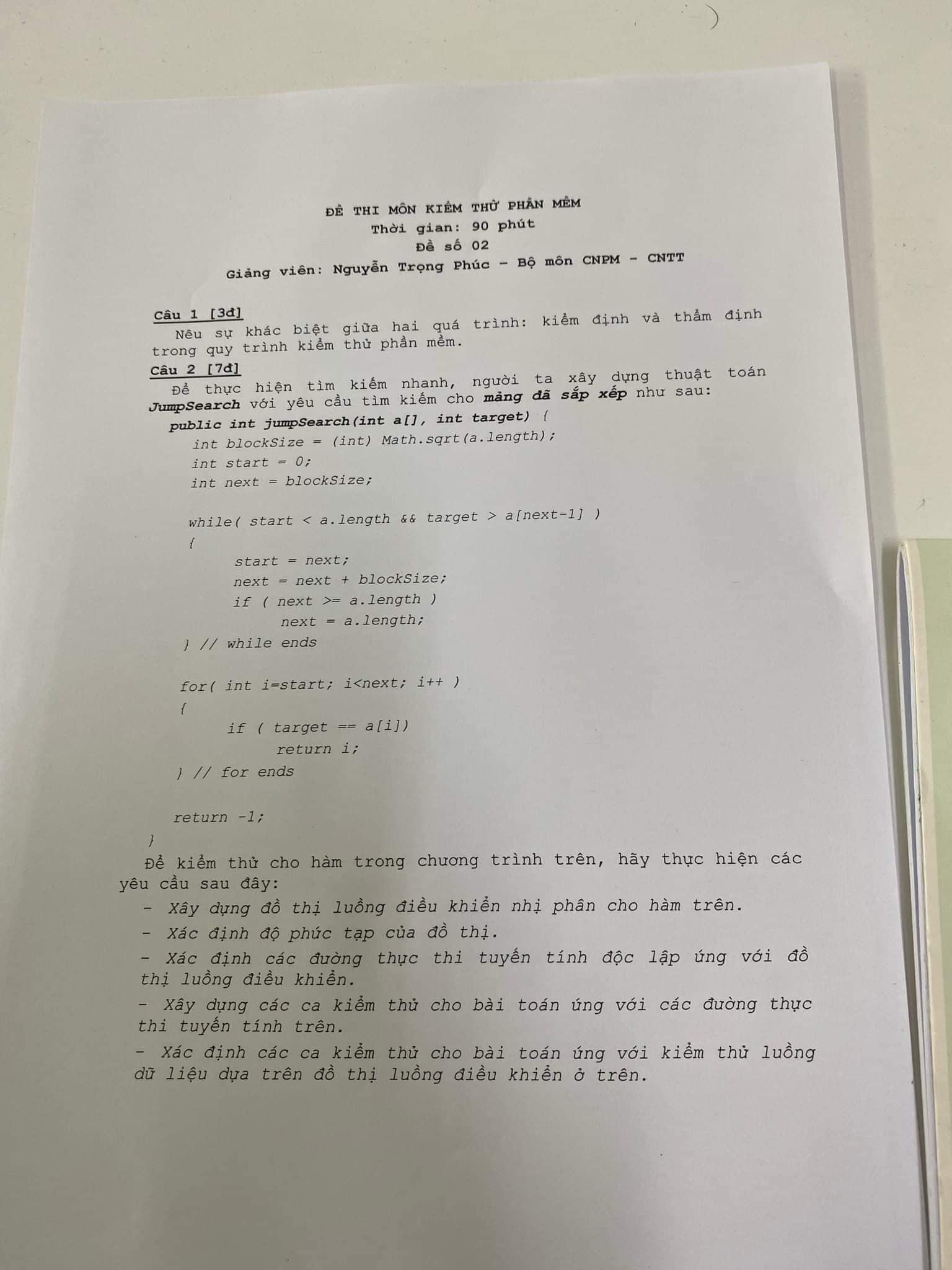
}

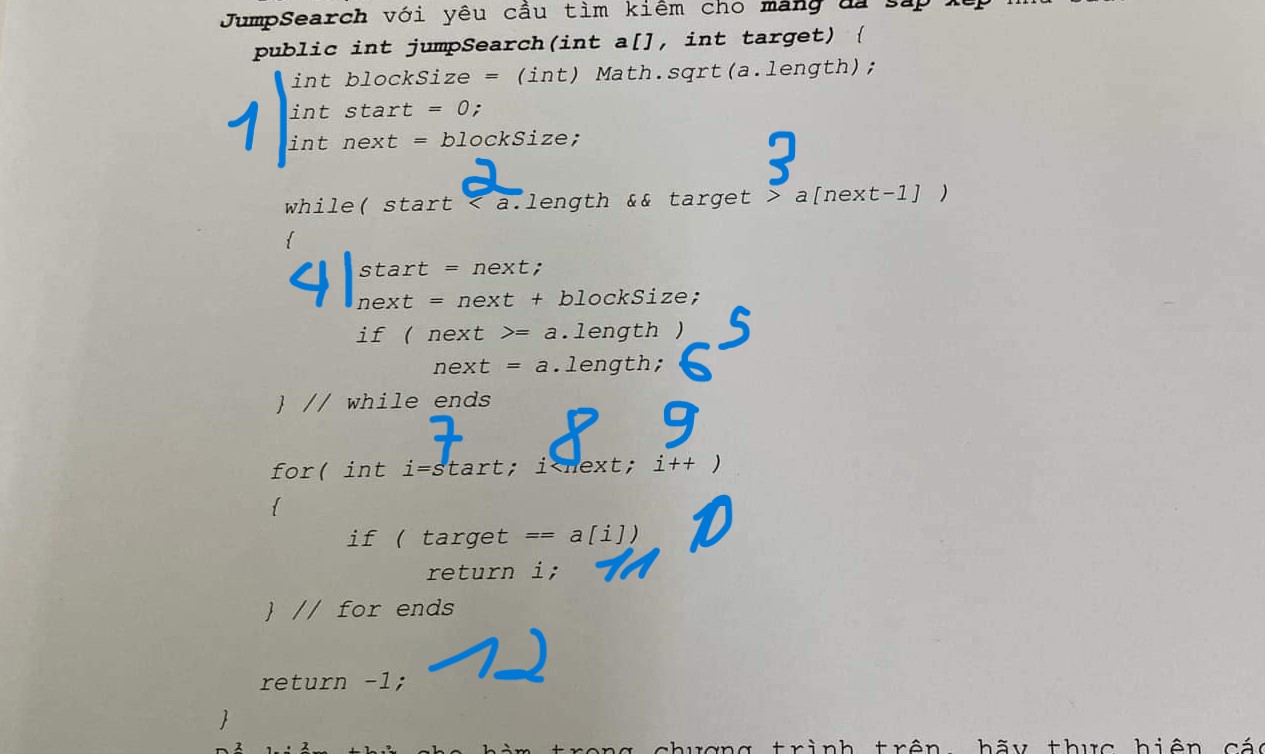
return a;

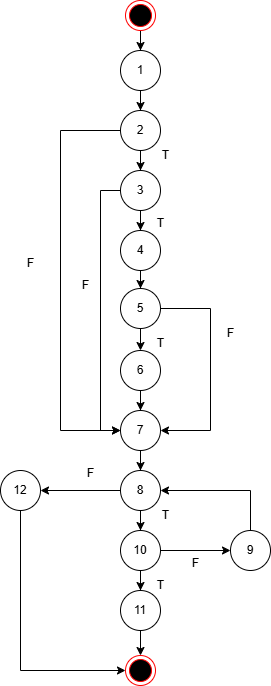
}

Để kiểm thử cho hàm trong chương trình trên, hãy thực hiện các yêu cầu sau đây:

* Xây dựng đồ thị luồng điều khiển nhị phân cho hàm trên.
* Xác định độ phức tạp của đồ thị.
* Xác định các đường thi hành tuyến tính độc lập ứng với đồ thị luồng điều khiển.
* Xác định các ca kiểm thử cơ bản.
* Dùng Junit thực hiện các ca kiểm thử trên.







Độ phức tap C = P +1= 5+1=6

Các đường:S->1->2->7->8->12->E

S->1->2->7->8->10->11->E

S->1->2->7->8->10->9->8->10->11->E

S->1->2->3->7->8->12->E

S->1->2->3->4->5->6->7->8->10->11->E

S->1->2->3->4->5->7->8->12->E

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test path | Input | expected |  |  |
|  | 1->2->7->8->12 | [],1 | -1 |  |  |
|  | 1->2->7->8->10->11 |  |  |  |  |
|  | 1->2->7->8->10->9->8->10->11 | [2,1,2],1 | 1 |  |  |
|  | 1->2->3->7->8->12 |  |  |  |  |
|  | 1->2->3->4->5->6->7->8->10->11 | [2,1,3],2 | 0 |  |  |
|  | 1->2->3->4->5->7->8->12 |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| D | U | K |
| A[],target |  |  |
| blockSize | A[] |  |
| Start |  |  |
| Next | blockSize |  |
|  | Start,a[], target, |  |
| start | Next |  |
| Next | Next, blockSize |  |
|  | Next, a[] |  |
| Next | A[] |  |
| i | Start, next |  |
| Target | A[] |  |
|  | i | i |
|  |  | -1 |

**Kiểm thử hộp đen:**

# Kĩ thuật phân lớp tương đương:

* Xác định các phân vùng tương thích
* Xác định số lượng kiểm tra cần thiết.
* Thiết kế Testcase.
* Thực thi Testcase
* So sánh và đánh giá về trạng thái, giá trị đầu ra mong muốn.
* Số lượng ca kiểm thử mong muốn
  + Custom name C1 = 3 [2<= number of characters <= 64]
  + Account number C2 = 3 [100000 <= account number <= 999999]
  + Loan amount C3 = 3 [500 <= amount <= 9000]
  + Term of loan C4 = 3 [1 <= term <= 30]
  + Monthly repayment C5 = 2 [repayment >= 10]
  + Số lượng ca kiểm thử: 3 \* 3 \* 3 \* 3 \* 2 = 162 ca kiểm thử.
* Số lượng ca kiểm thử thực tiễn: 10
  + Xây dựng 1 ca kiểm thử cho các trường hợp đúng.
  + Xây dựng n ca kiểm thử không hợp lệ ứng với mỗi biến.

C1 chia thành 3 vùng <2, 2->64, >64 trường hợp đúng là từ 2->64, trường hợp không hợp lệ là <2 và >64 nên ta có 3 TH kiểm thử trong đó 1 trường hợp đúng cà 2 trường hợp ko hợp lệ trương tự.

🡺 1 + 2 + 2 + 2 + 2 + 1 = 10.

# Kĩ thuật phân tích giá trị biên:

* Số lượng ca kiểm thử theo phân lớp tương đương
  + Custom name C1 = 3 [2<= number of characters <= 64]
  + Account number C2 = 3 [100000 <= account number <= 999999]
  + Loan amount C3 = 3 [500 <= amount <= 9000]
  + Term of loan C4 = 3 [1 <= term <= 30]
  + Monthly repayment C5 = 2 [repayment >= 10]
* Số lượng biên ứng với mỗi biến:
  + Custom name B1 = 2 => 6 ca kiểm thử
  + Account number B2 = 2 => 6 ca kiểm thử
  + Loan amount B3 = 2 => 6 ca kiểm thử
  + Term of loan B4 = 2 => 6 ca kiểm thử
  + Monthly repayment B5 = 1 => 3 ca kiểm thử

Số lượng ca kiểm thử thực tiên: 20

Trùng với biên = 2

Biên trên và biên dưới = 4 + 4 + 4 + 4 + 2 = 18

C1 có 2 biên trên + 2 biên dưới = 4 : 1,3,63,65

C5 có 1 biên trên + 1 biên dưới =2 = 9,11

* **Biên + phân lớp dữ liệu(phân lớp tương đương)**

Valid = 4 bao gồm chính giá trị biên =2 + 2 giá trị gần biên trong miền thỏa mãn

Ví dụ: C1 = 2,3,63,64

Invalid = 2+2+2+2+1=9 bao gồm 2 giá trị gần biên bên ngoài miền thỏa mãn

Ví dụ C1: 1,65

C5 : 9

**50 1m 1m4 2m**

**A1 : <50**

**A2: 50-1m**

**A3: 1m-1m4**

**A4: 1m4-2m**

**A5: >=2m**

**Valid : A2:3 + A3: 3 + A4: 3 = 9**

**Invalid : 2**

# Kỹ thuật dùng bảng quyết định

* + Một cửa hàng sách áp dụng các chương trình khuyến mại cho khách hàng như sau:
    - Nếu mua hàng vào ngày sinh nhật sẽ được giảm 20% trên hóa đơn. A1
    - Nếu có thẻ khách hàng thân thiết sẽ giảm 15% trên hóa đơn:A2
    - Với đơn hàng > 2triệu sẽ giảm 30% trên hóa đơn. A3
    - Không áp dụng đồng thời các chương trình khuyến mại, nếu có nhiều ưu đãi áp dụng ưu đãi cao nhất.
  + Kết hợp các kỹ thuật để xây dựng bảng quyết định

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Condition** |  |  |  |  |  |  |  |  |
| A1 | T | F | F | T | T | F | T | F |
| A2 | F | T | F | T | F | T | T | F |
| A3 | F | F | T | F | T | T | T | F |
| **Action** |  |  |  |  |  |  |  |  |
| Discount: % | 20 | 15 | 30 | 20 | 30 | 30 | 30 | 0 |

# Kỹ thuật dùng bảng chuyển trạng thái

# A diagram of a payment process Description automatically generated

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Trạng thái/ sự kiện | Give  Infor  start  paytimer | Pay timer expires | Cancel | Paid | Cancel/  Refund | Print/  Ticket | Cancel/(refund ticket)  Refund | Give ticket |
| S0-Start | S1 |  |  |  |  |  |  |  |
| S1- Made |  | S2 | S3 | S4 |  |  |  |  |
| S2-Cancelled  Nonpay |  |  |  |  |  |  |  |  |
| S3-Cancelled  bycust |  |  |  |  |  |  |  |  |
| S4-Paid |  |  |  |  | S3 | S5 |  |  |
| S5-Ticked |  |  |  |  |  |  | S3 | S6 |
| S6-Used |  |  |  |  |  |  |  |  |

# Kỹ thuật dựa trên lược đồ quan hệ nhân quả

* Để tính thuế thu nhập, người ta có mô tả sau:
  + Người vô gia cư nộp 4% thuế thu nhập.
  + Người có nhà ở nộp thuế theo bảng sau:
    - Nếu tổng thu nhập <=5.000.000 đồng thì chịu thuế 4%
    - Nếu tổng thu nhập >5.000.000 đồng thì chịu thuế 6%
* Yêu cầu: Áp dụng kỹ thuật lược đồ nhân quả trong kiểm thử hộp đen, hãy:
  + Xây dựng lược đồ nhân quả.
  + Xác định ma trận kiểm thử cho phép kiểm thử chức năng này.

C1: người vô gia cư

C2: Người có nhà ở

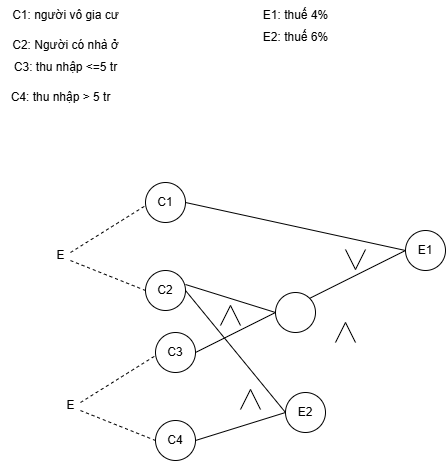
C3: thu nhập <=5 tr

C4: thu nhập > 5 tr

E1: thuế 4%

E2: thuế 6%

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **1** | **2** | **3** |
| **C1** | **1** | **0** | **0** |
| **C2** | **0** | **1** | **1** |
| **C3** | **0** | **1** | **0** |
| **C4** | **0** | **0** | **1** |
| **E1** | **1** | **1** |  |
| **E2** |  |  | **1** |

****

1. **D**
2. **D**
3. **D**
4. **D**