KIỂM THỬ LUỒNG DỮ LIỆU

Nguyễn Văn Huy - 18020651

1. **Trình bày các bước trong quy trình kiểm thử dòng dữ liệu động**
    i. Bước 1: Biểu diễn chương trình thành đồ thị
    ii. Bước 2: Lựa chọn tiêu chí kiểm thử
        + Với dòng điều khiển
        + Với dòng dữ liệu
    iii. Bước 3: Xác định đường đi thoả mãn tiêu chí
    iv. Bước 4: Xác định các ca kiểm thử
2. **CalFactorial**

7. Cho hàm `calFactorial` viết bằng ngôn ngữ C như Đoạn mã 7.7.

    - Hãy liệt kê các câu lệnh ứng với các khái niệm def, $c-use$, và $p-use$ ứng với các biến được sử dụng trong hàm này.
    - Hãy vẽ đồ thị dòng dữ liệu của hàm này.

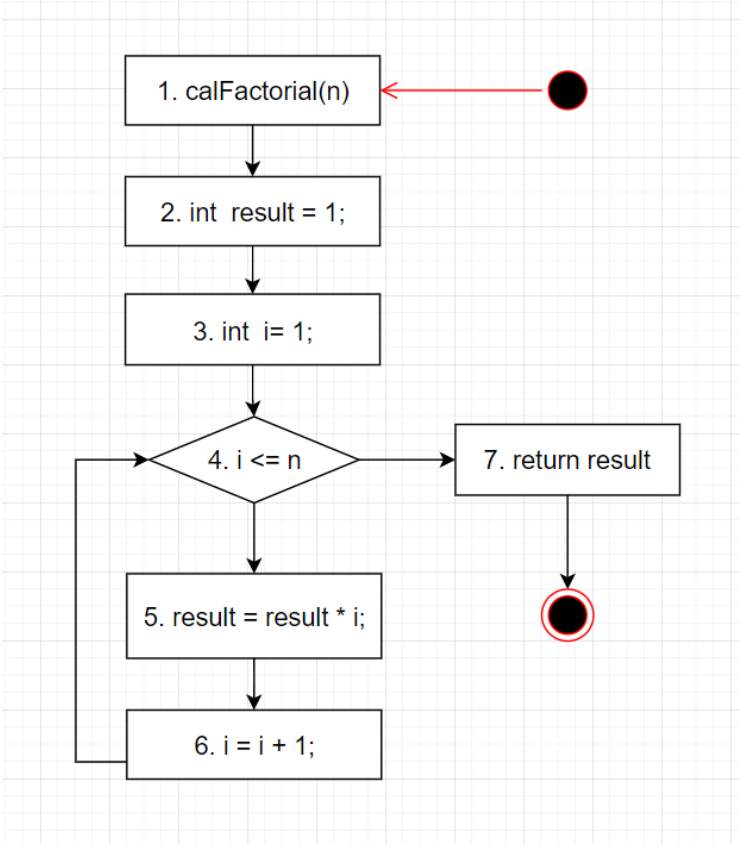
Đoạn mã 7.7: Mã nguồn C của hàm calFactorial

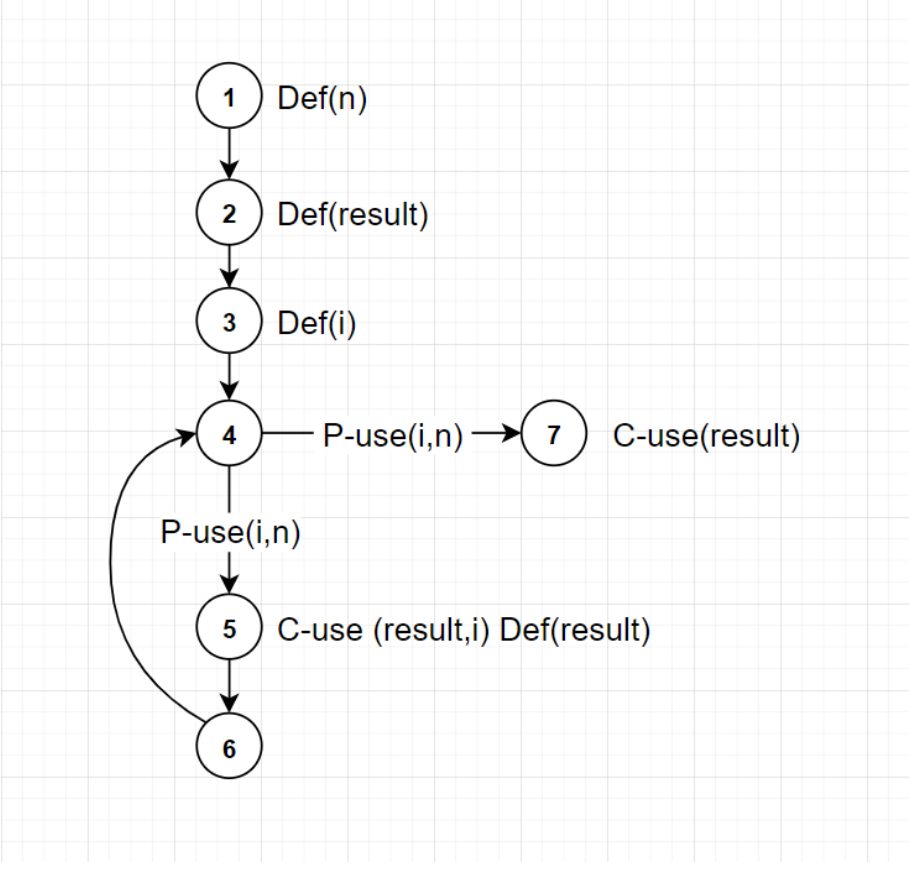
```
① int calFactorial (int n){
②         int result = 1;
③         int i=1;
④         while (i <= n){
⑤                 result = result *i;
⑥                 i++;
          }//end while
⑦         return result;
  }//the end
```

a. Liệt kê các câu lệnh:

Biến	def	c-use	p-use
n	(1)	-	(4)
result	(2), (5)	(5), (7)	-
i	(3), (6)	(5), (6)	(4)

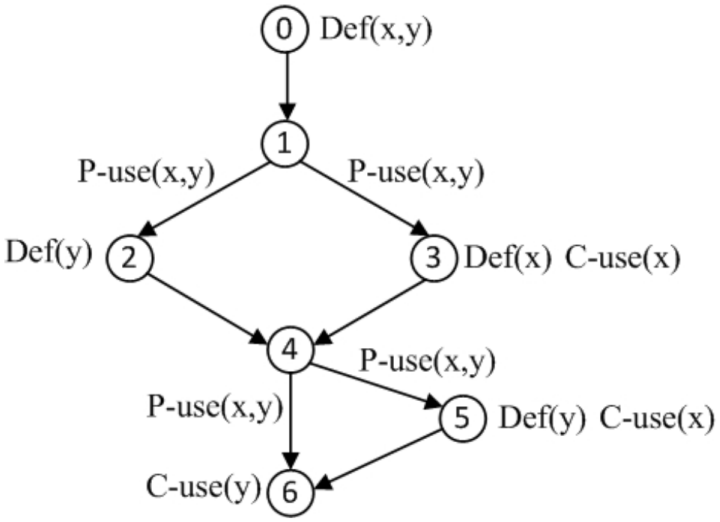
b. Vẽ đồ thị dòng dữ liệu:





3. Bài số 3

- Hãy xác định tất cả các *Def-clear-path* của các biến x và y.
- Hãy xác định tất cả các du-paths của các biến x và y.
- Hãy xác định tất cả các *All-p-uses/Some-c-uses* và *All-c-uses/Some-p-uses* (dựa vào các chuẩn của kiểm thử dòng dữ liệu).
- Biểu thức của các *p-use*(x, y) tại cạnh (1,3) và (4,5) lần lượt là $x + y = 4$ và $x^2 + y^2 > 17$. Đường đi (0 - 1 - 3 - 4 - 5 - 6) có thực thi được không? Giải thích.
- Tại sao tại đỉnh 3 biến x được định nghĩa và sử dụng nhưng không tồn tại mối quan hệ def-use?



a. Xác định tất cả các Def-clear-path của biến x và y:

Def(x)={0,3}, Use(x)={1,3,4,5}, P-use(x)={1,4}, C-use(x)={3,5}

Def(y)={0,2,5}, Use(y)={1,4,6}, P-use(y)={1,4} C-use(y)={6}

Biến	Def-clear-path	du-path
x	0 - 1 0 - 1 - 3 0 - 1 - 2 0 - 1 - 2 - 4 0 - 1 - 2 - 4 - 5 0 - 1 - 2 - 4 - 5 - 6 0 - 1 - 2 - 4 - 6 3 - 4 3 - 4 - 5 3 - 4 - 5 - 6 3 - 4 - 6	0 - 1 0 - 3 0 - 4 0 - 5 3 - 4 3 - 5
y	0 - 1 0 - 1 - 3 0 - 1 - 3 - 4 0 - 1 - 3 - 4 - 5 0 - 1 - 3 - 4 - 6 2 - 4 2 - 4 - 6 2 - 4 - 5 5 - 6	0 - 1 0 - 4 0 - 6 2 - 4 2 - 6 5 - 6

b. Xác định tất cả các All-p-uses/Some-c-uses và All-c-uses/Some-p-uses

i. All-p-uses/Some-c-uses

Biến	Du-pair	Def-clear path	Complete path
x	0 - 1	0 - 1	0 - 1 - 2 - 4 - 6
	0 - 4	0 - 1 - 2 - 4	
	3 - 4	3 - 4	0 - 1 - 3 - 4 - 6
y	0 - 1	0 - 1	0 - 1 - 3 - 4 -6
	0 - 4	0 - 1 - 3 - 4	
	2 - 4	2 - 4	0 - 1 - 2 - 4 - 5 -6
	5 - 6	5 - 6	

ii. All-c-uses/Some-p-uses

Biến	Du-pair	Def-clear path	Complete path
x	0 - 3	0 - 1 - 3	0 - 1 - 3 - 4 - 5 -6
	3 - 5	3 - 4 - 5	
	0 - 5	0 - 1 - 2 - 4 - 5	0 - 1 - 2 - 4 - 5 -6
y	0 - 6	0 - 1 - 3 - 4 - 6	0 - 1 - 3 - 4 - 6
	2 - 6	2 - 4 - 6	0 - 1 - 2 - 4 - 6
	5 - 6	5 - 6	0 -1 - 2 - 4 - 5 - 6

c. Biểu thức của các p-use(x; y) tại cạnh (1,3) và (4,5) lần lượt là x + y = 4 và $x^2 + y^2 > 17$. Đường đi (0 - 1 - 3 - 4 - 5 - 6) có thực thi được không? Giải thích.

Trả lời: Có thể được thực thi vì tại nút 3, biến x được gán lại giá trị mới và có thể sẽ thỏa mãn điều kiện tại cạnh (4,5).

d. Tại sao tại đỉnh 3 biến x được định nghĩa và sử dụng nhưng không tồn tại mối quan hệ def-use?

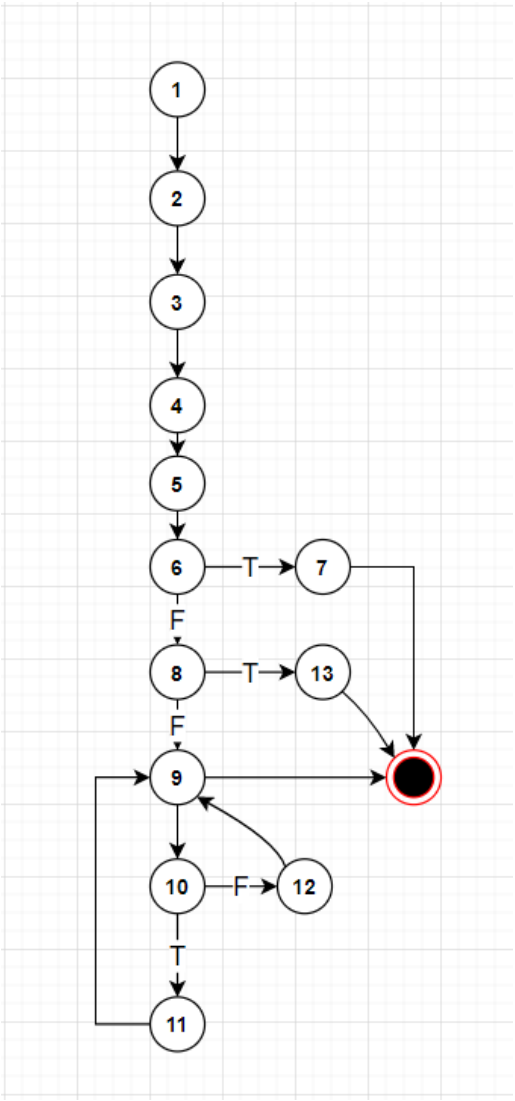
Trả lời: Vì câu lệnh use được thực thi trước câu lệnh def.

4. Ước chung lớn nhất

```
int UCLN(int m, int n){
        if (m < 0) m = -m;
        if (n < 0) n = -n;
        if (m == 0)    return n;
        if (n == 0)    return m;
        while (m != n) {
                if(m > n)
                        m = m - n;
                else
                        n = n - m;
        }//end while
        return m;
}
```

a. Xây dựng CFG cho hàm UCLN

```
int UCLN (int m, int n){ 1
  if (m < 0) {           2
    m = -m;              3
  }
  if (n < 0) {           4
    n = -n;              5
  }
  if (m == 0) {          6
      return n;          7
    }
  if (n == 0) {          8
    return m;            13
  }
  while (m != n) {       9
    if(m > n)            10
        m = m – n;       11
    else
        n = n – m;       12
  }// end while
  return m;              13
}
```



b. Sinh đường đi và các ca kiểm thử với độ đo C2

UCLN(0, 0): 1 - 2(F) - 4(F) - 6(T) - 7

UCLN(1, 0): 1 - 2(F) - 4(F) - 6(F) - 8(T) - 13

UCLN(3, 2): 1 - 2(F) - 4(F) - 6(F) - 8(F) - 9(T) - 10(T) - 11 - 9(T) - 10(F) - 12 - 9(F) - 13

c. Sinh đường đi và các ca kiểm thử với độ đo all-def coverage

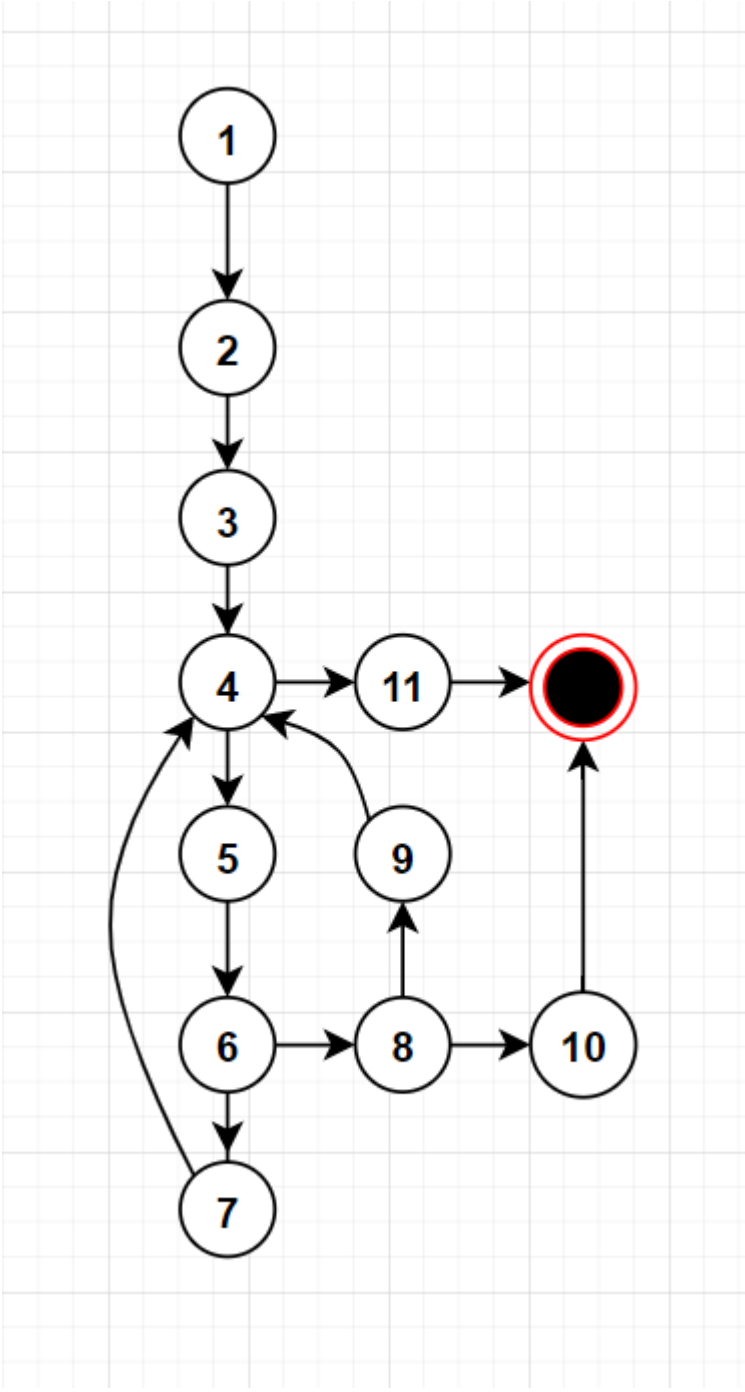
Biến	Du-pair	Def-clear path	Complete path	Ca kiểm thử (m,n)
m	1 - 2 3 - 6 12 - 10	1 - 2 3 - 4(T) - 5 - 6(F) 3 - 4(F) - 6(F)	1 - 2 - 3 - 4(T) - 5 - 6(F) - 8 (F) - 9(T) - 13 1 - 2 - 3 - 4(F) - 6(F) - 8(F) - 9(T) - 13	(-3, -3) (3, 3)
n	1 - 2 5 - 8 14 - 10	1 - 2(T) 1 - 2(F) 5 - 6(T) - 7 - 8(F) 5 - 6(F) - 8(F)	1 - 2(F) - 4(T) - 5 - 6(T) - 7 - 8(F) 1 - 2(F) - 4(T) - 5 - 6(F) - 8(F) - 9(F) - 13	(-3, 3)

5. BinSearch

```
1  int Binsearch(int x, int v[], int n){
2          int low = 0, high, mid;
3          high = n - 1;
4          while (low <= high) {
5                  mid = (low + high)/2;
6                  if (x < v[mid])
7                          high = mid - 1;
                   else
8                          if (x > v[mid])
9                                  low = mid + 1;
                           else
10                                 return mid;
           }//end while
11         return -1;
   }//the end
```

- Xây dựng đồ thị luồng điều khiển cho hàm BinSearch
- Sinh các đường đi và các ca kiểm thử với độ đo C2.
- Liệt kê các cặp du-pairs của tất cả các biến trong chương trình
- Sinh các đường đi và các ca kiểm thử với độ đo All-def cho biến **high**
- Sinh các đường đi và các ca kiểm thử với độ đo All-p-use cho biến **x**

a. Xây dựng đồ thị luồng điều khiển



b. Sinh đường đi và các ca kiểm thử với độ đo C2

Ca kiểm thử (x , v[], n)	Đường đi
(1, [0, 1], 2)	1 - 2 - 3 - 4(T) - 5 - 6(F) - 8(T) - 9 - 4 - 5 - 6(F) - 8(F) - 10
(0, [0, 1, 2], 3)	1 - 2 - 3 - 4(T) - 5 - 6(T) - 7 - 4(T) -5 - 6(F) - 8(F) - 10
(0, [1], 1)	1 - 2 - 3 - 4(T) - 5 - 6(T) - 7 - 4(F) - 11

c. Liệt kê Du Pair

Biến	Du-pair
x	1 - 6 1 - 8
v	1 - 6 1- 8
n	1 - 3
low	2 - 4 2 - 5 9 - 4 9 - 5
high	2 - 4 2 - 5 7 - 4

	7 - 5
mid	5 - 6 5 - 8 5 - 7 5 - 9 5 - 10

d. Đường đi và ca kiểm thử cho All-def của biến high

Biến	Du pair	Def-clear-path	Complete path	Ca kiểm thử
high	2 - 4(T) 2 - 4(F) 2 - 5 7 - 4(T) 7 - 4(F) 7 - 5	2 - 3 - 4(T) 2 - 3 - 4(F) 2 - 3 - 4(T) - 5 7 - 4(T) 7 - 4(F) 7 - 5	1 - 2 - 3 - 4(T) 1 - 2 - 3 - 4(F) 1 - 2 - 3 - 4(T) - 5 1 - 2 - 3 - 4(T) - 5 - 6(T) - 7 - 4(T) 1 - 2 - 3 - 4(T) - 5 - 6(T) - 7 - 4(T) - 5	(0, [0, 1, 2], 3) (0, [], 0)

e. Đường đi và ca kiểm thử cho All-p-use của biến x

Biến	Du pair	Def-clear path	Complete path	Ca kiểm thử
x	1 - 6 1 - 8	1- 2 - 3 - 4(T) - 5 - 6(T) 1- 2 - 3 - 4(T) - 5 - 6(F) 1- 2 - 3 - 4(T) - 5 - 6(F) - 8(T) 1- 2 - 3 - 4(T) - 5 - 6(F) - 8(F)	1- 2 - 3 - 4(T) - 5 - 6(T) - 7 - 4(T) - 5 - 6(F) - 8(F) - 10 1- 2 - 3 - 4(T) - 5 - 6(F) - 8(F) - 10 1- 2 - 3 - 4(T) - 5 - 6(F) - 8(T) - 9 - 4(T) - 5 - 6(F) - 8(F) - 10	(0, [0, 1, 2], 3) (1, [0, 1], 2) (2, [0, 1, 2], 3)

6. Kiểm thử chương trình của bạn với độ phủ All-c-uses/some-puses

a. Bài toán: *Nhập tháng và năm (max: 9999), cho biết tháng đó có bao nhiêu ngày.*

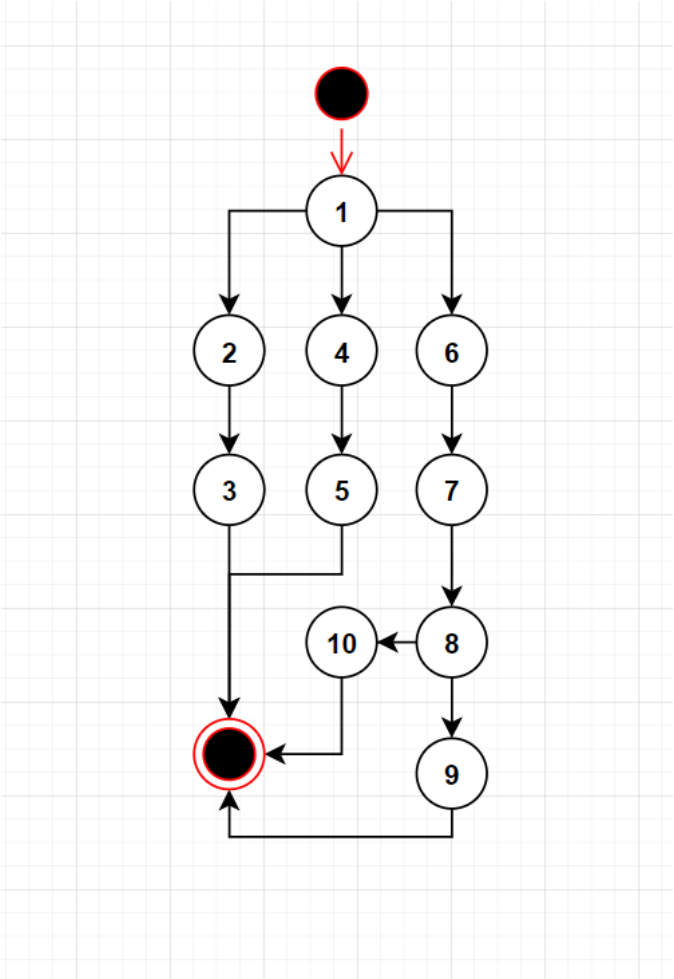
b. Mã nguồn:

```
// Nhập tháng và năm => Cho biết tháng đó có bao nhiêu ngày

const assert = require('assert');
const isLeapYear = (year) => {
  return (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;
};

const countDays = (month, year) => { 0
1 switch (month) {
2   case 1: case 3: case 5: case 7: case 8: case 10: case 12:
3     return 31;
      break;
4   case 4: case 6: case 9: case 11:
5     return 30;
      break;
6   case 2:
7     let check = isLeapYear(year);
8     if (check == 1) {
9       return 29;
      } else {
10      return 28;
      }
  }
};
```

c. *Xây dựng đồ thị dòng điều khiển :*



d. Sinh ca kiểm thử với độ phủ All-c-uses/some-puses

Biến	Def	C-use	P-use
month	(0)	-	(1)
year	(0)	(7)	-
check	(7)	-	(8)

Biến	Du-pair	Def-clear path	Complete path	Ca kiểm thử (month,year)
month	0 - 1	0 - 1	0 - 1 - 6 -7 - 8 -10	(02, 2021)
year	0 - 7	0 - 6 -7		
check	7 - 8	7 - 8		