Câu 7:

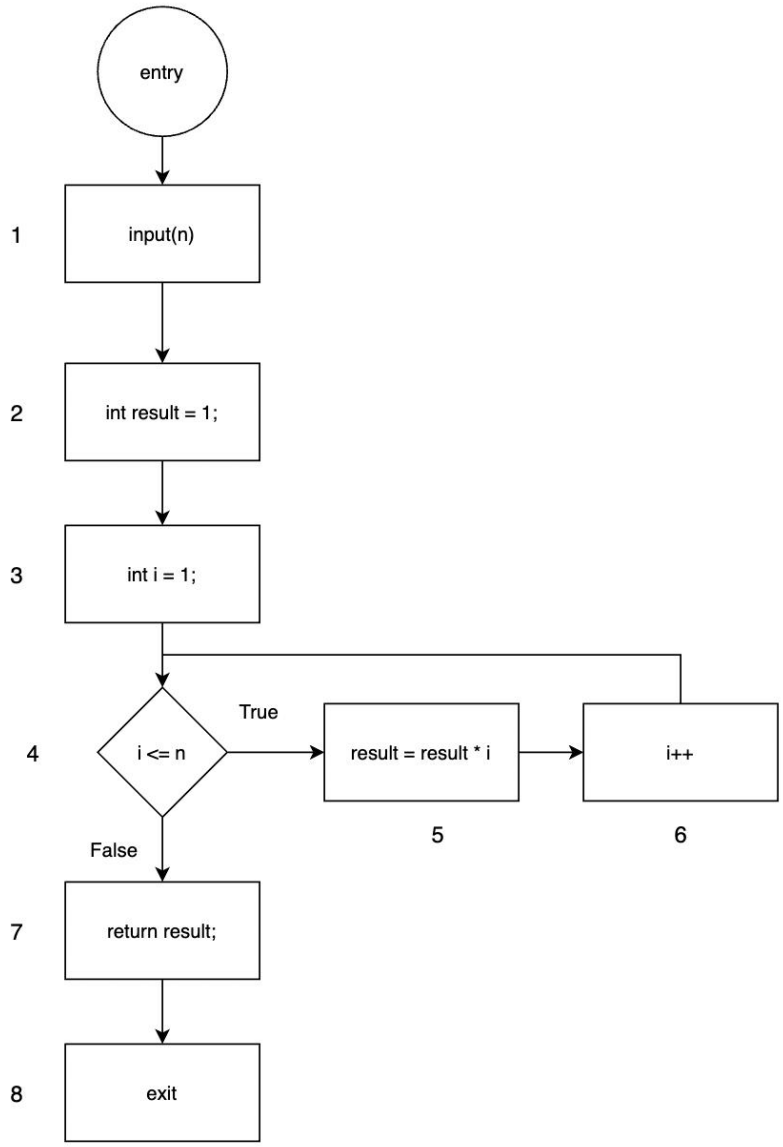
7. Cho hàm `calFactorial` viết bằng ngôn ngữ C như Đoạn mã 7.7.

   - 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

```
int calFactorial (int n){
        int result = 1;
        int i=1;
        while (i <= n){
                result = result *i;
                i++;
        }//end while
        return result;
}//the end
```

- Đồ thị dòng dữ liệu:



- Các đại lượng:

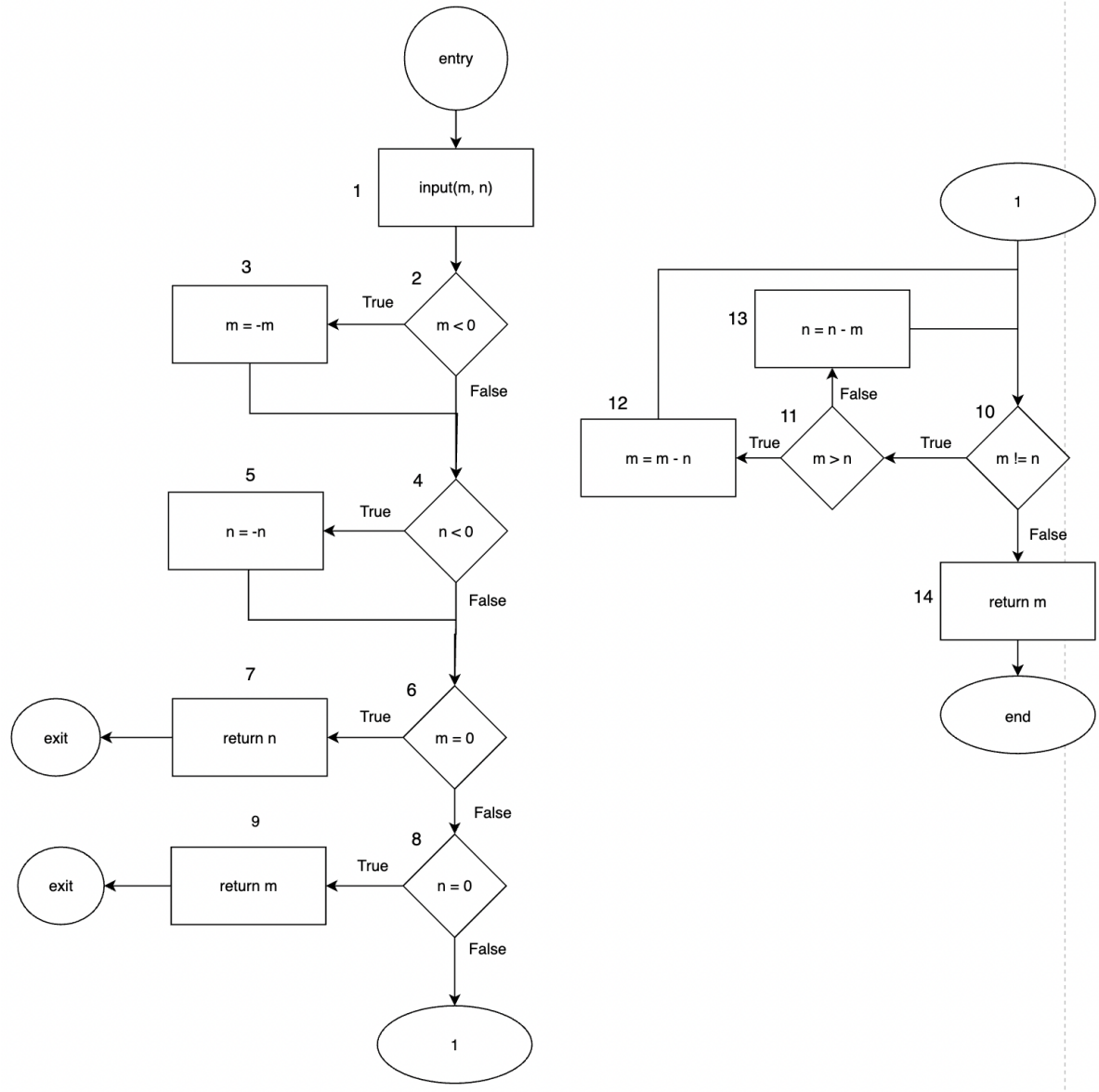
- Def(n): 1; c-use(n): không có; p-use(n): 4.
- Def(result): 2, 5; c-use(result): 5, 7; p-use(result): không có.
- Def(i): 3, 6; c-use(i): 5, 6; p-use(i): 4

Bài 3 UCLN:

Đoạn mã 6.4: Mã nguồn của hàm UCLN

```
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;
}
```

1. **CFG cho hàm UCLN với đồ thị C2**



2. **Đường đi và các ca kiểm thử độ đo C2**

Test ID	Test path	Test case

1	1,2(F),4(F), 6(F), 8(F), 10(T), 11(T), 12, 10(F), 14	m=4, n=2
2	1,2(T),3,4(T), 5, 6(F), 8(F), 10(F), 14	m=-2, n=-2
3	1,2(F),4(F), 6(T), 7	m=0, n=2
4	1,2(F),4(F), 6(F), 8(T), 9	m=4, n=0
5	1,2(F),4(F), 6(F), 8(F), 10(T), 11(F), 13, 10(F), 14	m=2, n=4

3. **Đường đi và các ca kiểm thử độ đo all-def coverage**

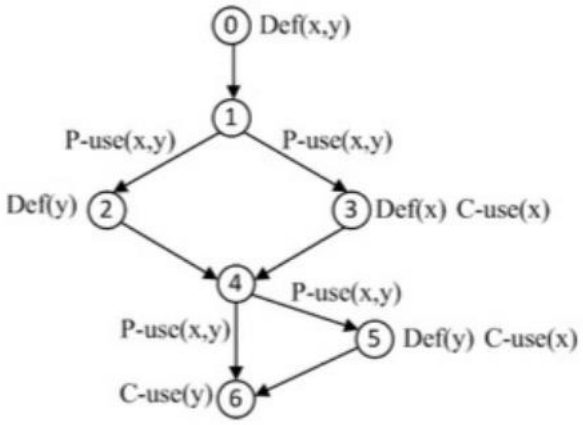
def(m): 1, 3, 12

use(m): 2,3,6,9,10,11,12,13,14

def(n): 1, 5, 13

	Du-pair	Def-clear-path	Complete Path	Test case
m	1,3	1,2(T),3	1,2(T),3,4(T), 5, 6(F), 8(F), 10(F), 14	m=-2, n=-2
	3,6	3,4(T), 5, 6(F)	1,2(T),3,4(T), 5, 6(F), 8(F), 10(F), 14	m=-2, n=-2
	12,10	12, 10(F), 14	1,2(F),4(F), 6(F), 8(F), 10(T), 11(T), 12, 10(F), 14	m=4, n=2
n	1,5	1,2(T),3,4(T), 5	1,2(T),3,4(T), 5, 6(F), 8(F), 10(F), 14	m=-2, n=-2
	5,7	5, 6(T), 7	1,2(F),4(T),5, 6(T), 7	m=0, n=-2
	13,10	13,10	1,2(F),4(F), 6(F), 8(F), 10(T), 11(F), 13, 10(F), 14	m=2, n=4

Câu 10:



- Def-clear-path của x:
  - (0,1,2), (0,1), (0,1,3), (0,1,2,4), (0,1,2,4,5), (0,1,2,4,6), (0,1,2,4,5,6)
  - (3,4,6), (3,4), (3,4,5), (3,4,5,6)
- Def-clear-path của y:
  - (0,1,3,4), (0, 1), (0,1,3), (0,1,3,4,6)
  - (2,4), (2,4,6)
  - (5,6)
- Du-paths:

	Du-pair
x	0,1
	0,3
	0,4
	0,5
	3,4
	3,5
y	0,1
	0,4
	0,6
	2,4
	2,6
	5,6

- All-p-uses/ Some-c-uses:

	Du-pair
x	0,1(T)
	0,1(F)

	0,4(T)
	0,4(F)
	3,1
	3,4(T)
	3,4(F)
y	0,1(T)
	0,1(F)
	0,4(T)
	0,4(F)
	2,4(T)
	2,4(F)
	5,6

- All-c-uses/some-p-use:

	Du-pair	Def-clear-path	Complete Path
x	0,3	0,1,3	0,1,3,4,5,6
	0,5	0,1,2,4,5	0,1,2,4,5,6
	3,3	không có	
	3,5	3,4,5	0,1,3,4,5,6
y	0,6	0,1	0,1,2,4,5,6
	2,6	2,4	0,1,2,4,5,6
	5,6	5,6	0,1,2,4,5,6

- Nếu biểu thức p-use(x,y) tại cạnh (1,3) và (4,5) lần lượt là x+y=4 và $x^2+y^2>17$ thì đường đi (0-1-3-4-5-6) không được thực thi. Tại vì:
  - x^2+y^2 < hoặc = $(x+y)^2 = 16$ trong khi giả thiết $x^2+y^2>17$. Điều này là vô lý.
- Đỉnh 3 biến x được định nghĩa và sử dụng nhưng không tồn tại mối quan hệ def-use bởi vì use(x) trước rồi mới def(x).

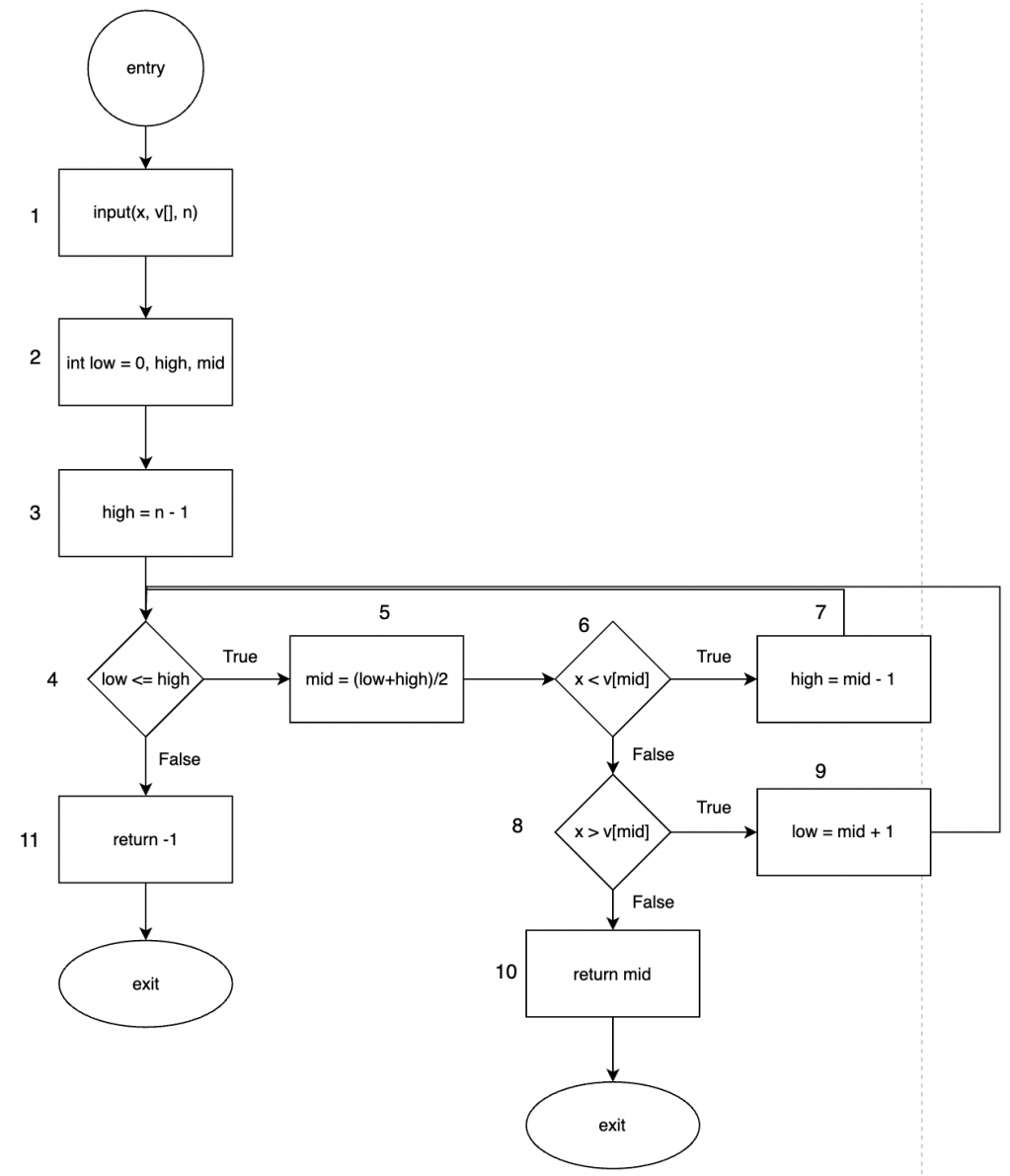
Bài 4:

Cho đoạn mã 6.2. Hãy:
- 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**

Đoạn mã 6.2: Mã nguồn của hàm 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
```

Đồ thị luồng điều khiển hàm BinSearch:



Đường đi và ca kiểm thử độ đo C2:

Các câu lệnh điều kiện: 4, 6, 8

Test ID	Test path	Test case
1	1,2,3,4(T),5,6(T),7,4(F),11	x=0,v=[1],n=1
2	1,2,3,4(T),5,6(F),8(T),9,4(F),11	x=2,v=[1],n=1
3	1,2,3,4(T),5,6(F),8(F),10,4(F),11	x=1,v=[1],n=1

Các cặp du-pairs của tất cả các biến:

	Du-pair	Path
x	1,6	1,2,3,4,5,6,7,4,11
	1,8	1,2,3,4,5,6,8,9,4,11
v	1,6	1,2,3,4,5,6,7,4,11
	1,8	1,2,3,4,5,6,8,9,4,11
n	1,3	1,2,3,4,5,6,8,9,4,11
low	2,4	1,2,3,4,5,6,8,9,4,11
	2,5	1,2,3,4,5,6,8,9,4,11
	9,4	1,2,3,4,5,6,8,9,4,11
	9,5	1,2,3,4,5,6,8,9,4,5,6,7,11
high	3,4	1,2,3,4,5,6,8,9,4,11

	3,5	1,2,3,4,5,6,8,9,4,11
	7,4	1,2,3,4,5,6,7,4,11
	7,5	1,2,3,4,5,6,7,4,5,6,7,11
mid	5,7	1,2,3,4,5,6,7,4,11
	5,9	1,2,3,4,5,6,8,9,4,11
	5,10	1,2,3,4,5,6,8,10,4,11

Đường đi và ca kiểm thử độ đo all-def cho biến high:

	Du-pair	Def-clear-path	Complete Path	Test case
high	3,4	3,4	1,2,3,4(T),5,6(T),7,4(F),11	x=0,v=[1],n=1
	7,4	7,4(F)	1,2,3,4(T),5,6(T),7,4(F),11	x=0,v=[1],n=1

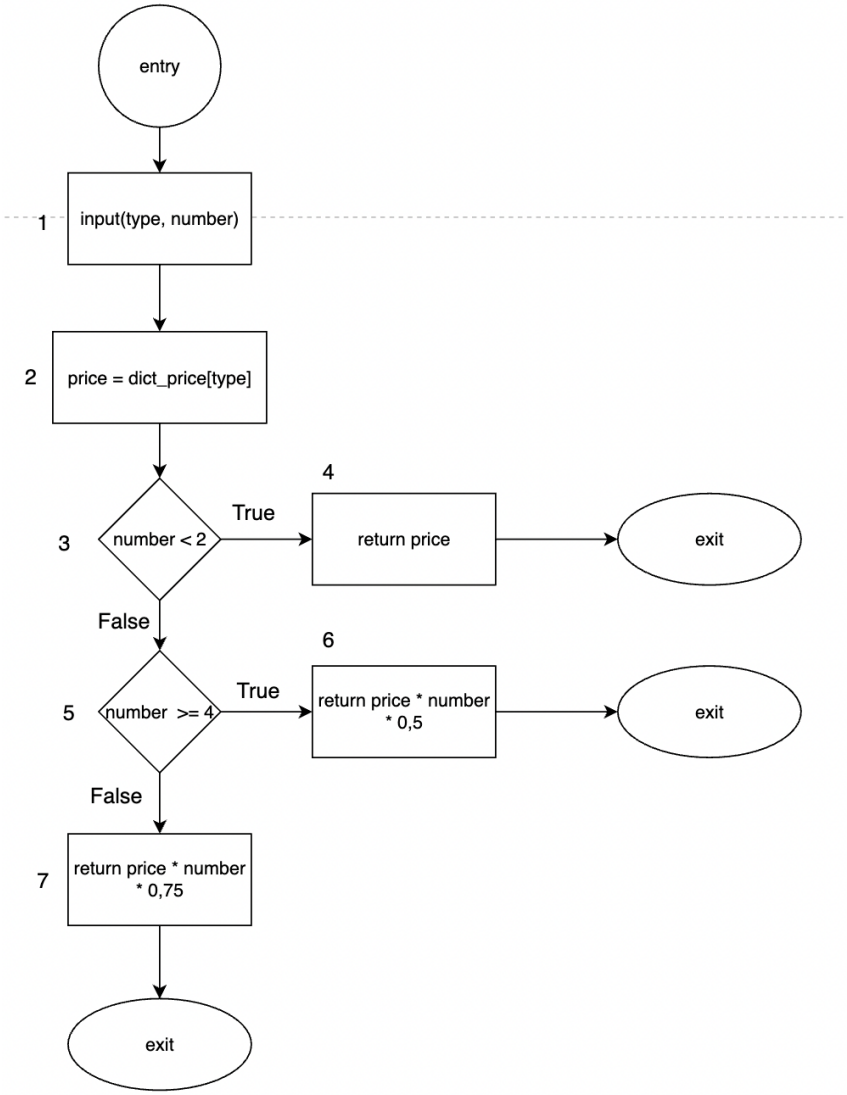
Đường đi và ca kiểm thử độ đo all-p-use cho biến x:

	Du-pair	Def-clear-path	Complete Path	Test case
x	1,6(F)	1,2,3,4(T),5,6(F)	1,2,3,4(T),5,6(F),8(T),9,4(F),11	x=2,v=[1],n=1
	1,6(T)	1,2,3,4(T),5,6(T)	1,2,3,4(T),5,6(T),7,4(F),11	x=0,v=[1],n=1
	1,8(F)	1,2,3,4(T),5,6(F),8(F)	1,2,3,4(T),5,6(F),8(F),10,4(F),11	x=1,v=[1],n=1
	1,8(T)	1,2,3,4(T),5,6(F),8(T)	1,2,3,4(T),5,6(F),8(T),9,4(F),11	x=2,v=[1],n=1

Chương trình tự viết với độ đo all-c-uses/some-p-uses

```
import unittest
dict_price = {
        1: 30000,
        2: 40000
    }
def count_money(type, number):
    price = dict_price[type]
    if number < 2:
        return price
    elif number >= 4:
        return price * number * 0.5
    else:
        return price * number * 0.75
```

Đồ thị CFG:



Đường đi và ca kiểm thử độ đo all-c-uses/some-p-uses:

	Du-pair	Def-clear-path	Complete Path	Test case	Expected Output	Actual Output	Status
type	1,2	1,2	1,2,3(F),5(T),6	type=1,number=4	60000	60000	True
number	1,6	1,2,3(F),5(T),6	1,2,3(F),5(T),6	type=1,number=4	60000	60000	True
	1,7	1,2,3(F),5(F),7	1,2,3(F),5(F),7	type=1,number=2	45000	45000	True
price	2,4	2,3(T),4	1,2,3(T),4	type=1,number=1	30000	60000	True
	2,6	2,3(F),5(T),6	1,2,3(F),5(T),6	type=1,number=4	60000	60000	True
	2,7	2,3(F),5(F),7	1,2,3(F),5(F),7	type=1,number=2	45000	45000	True

Code:

```
class TestStringMethods(unittest.TestCase):
    def test_flow_data_1(self):
        self.assertEqual(count_money(1, 1), 30000)

    def test_flow_data_2(self):
        self.assertEqual(count_money(1, 2), 45000)

    def test_flow_data_3(self):
        self.assertEqual(count_money(1, 4), 60000)

if __name__ == '__main__':
    unittest.main(verbosity=2)
```