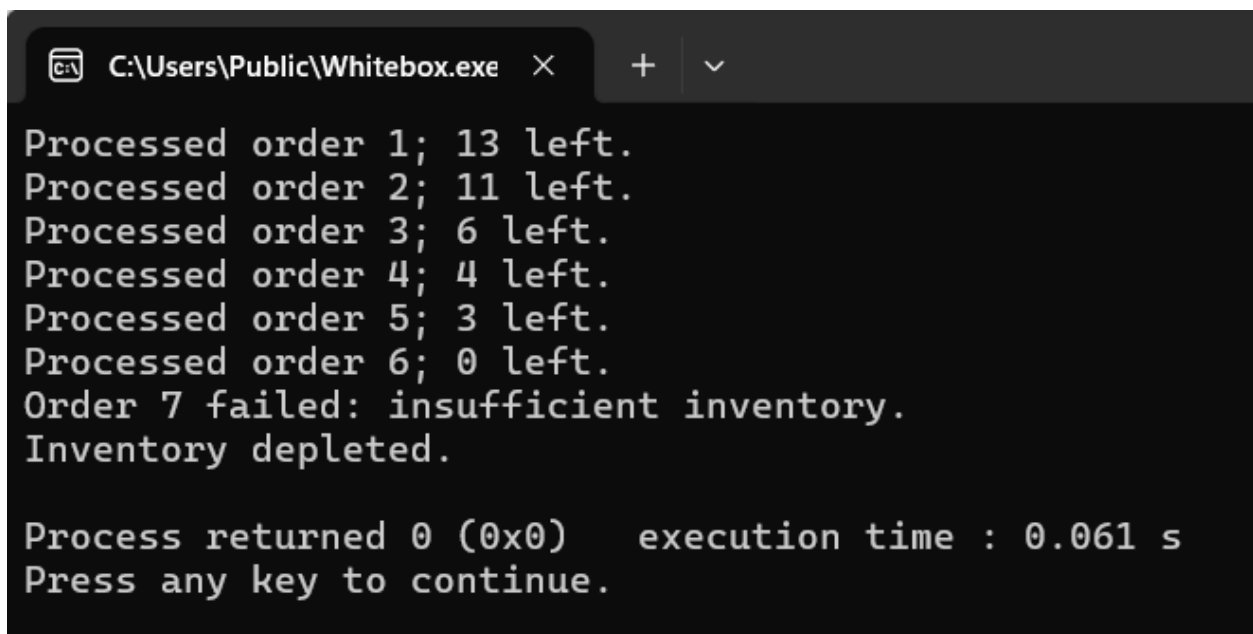


MSSV: 22521329	BÁO CÁO BÀI TẬP THỰC HÀNH TUẦN 4
Họ và tên: Nguyễn Cao Thắng	
Lớp: IE108.O21.CNVN.1	

Bài tập 2.1. Kiểm thử độ phủ câu lệnh (Statement Coverage – C0), độ phủ nhánh (Branch Coverage – C1) và độ phủ điều kiện (Condition Coverage – C2)

- Input với MSSV là 22521329:
 - orders = {2, 2, 5, 2, 1, 3, 2, 9};
 - inventory = 15;
 - threshold = 8;
- Kết quả thực hiện chương trình:



```

C:\Users\Public\Whitebox.exe
Processed order 1; 13 left.
Processed order 2; 11 left.
Processed order 3; 6 left.
Processed order 4; 4 left.
Processed order 5; 3 left.
Processed order 6; 0 left.
Order 7 failed: insufficient inventory.
Inventory depleted.

Process returned 0 (0x0)   execution time : 0.061 s
Press any key to continue.
  
```

- Giải thích kết quả:
 - Processed order 1; 13 left.
 - Đơn hàng thứ nhất yêu cầu 2 sản phẩm.
 - Vì inventory ban đầu là 15 và đủ hàng tồn kho, nên đơn hàng thứ nhất được thực hiện. inventory giảm xuống còn 13.
 - Processed order 2; 11 left.
 - Đơn hàng thứ hai yêu cầu 2 sản phẩm.
 - Vì inventory hiện tại là 13 và đủ hàng tồn kho, nên đơn hàng thứ hai được thực hiện. inventory giảm xuống còn 11.
 - Processed order 3; 6 left.
 - Đơn hàng thứ ba yêu cầu 5 sản phẩm.

- Vì inventory hiện tại là 11 và đủ hàng tồn kho, nên đơn hàng thứ ba được thực hiện. inventory giảm xuống còn 6.
- Processed order 4; 4 left.
 - Đơn hàng thứ tư yêu cầu 2 sản phẩm.
 - Vì inventory hiện tại là 6 và đủ hàng tồn kho, nên đơn hàng thứ tư được thực hiện. inventory giảm xuống còn 4.
- Processed order 5; 3 left.
 - Đơn hàng thứ năm yêu cầu 1 sản phẩm.
 - Vì inventory hiện tại là 4 và đủ hàng tồn kho, nên đơn hàng thứ năm được thực hiện. inventory giảm xuống còn 3.
- Processed order 6; 0 left.
 - Đơn hàng thứ sáu yêu cầu 3 sản phẩm.
 - Vì inventory hiện tại là 3 và đủ hàng tồn kho, nên đơn hàng thứ sáu được thực hiện. inventory giảm xuống còn 0.
- Order 7 failed: insufficient inventory.
 - Đơn hàng thứ bảy yêu cầu 2 sản phẩm.
 - Vì inventory hiện tại là 0 và không đủ hàng tồn kho, nên đơn hàng thứ bảy không thể thực hiện.
- Inventory depleted.
 - Sau khi xử lý tất cả các đơn hàng, inventory còn lại là 0, nên thông báo “Inventory depleted.”
- Độ phủ câu lệnh C0:

Mã nguồn	Đã thực thi?
void manageInventory(int inventory, ...	–
{	–
for (size_t i = 0; i < orders.size(); ++i) {	O
if (inventory >= orders[i]) {	O
inventory -= orders[i];	O
cout << "Processed order " << i + 1 ...	O
} else {	–
cout << "Order " << i + 1 << " failed...	O
break;	O
}	–
}	–
if (inventory == 0) {	O
cout << "Inventory depleted." << endl;	O
} else if (inventory <= threshold) {	X (inventory = 0)
cout << "Low inventory: " << inventory ...	X
} else {	–

cout << "Inventory sufficient: " << ...	X
}	–
}	–

Độ phủ câu lệnh $C0 = \frac{9}{11} \approx 81.8\%$

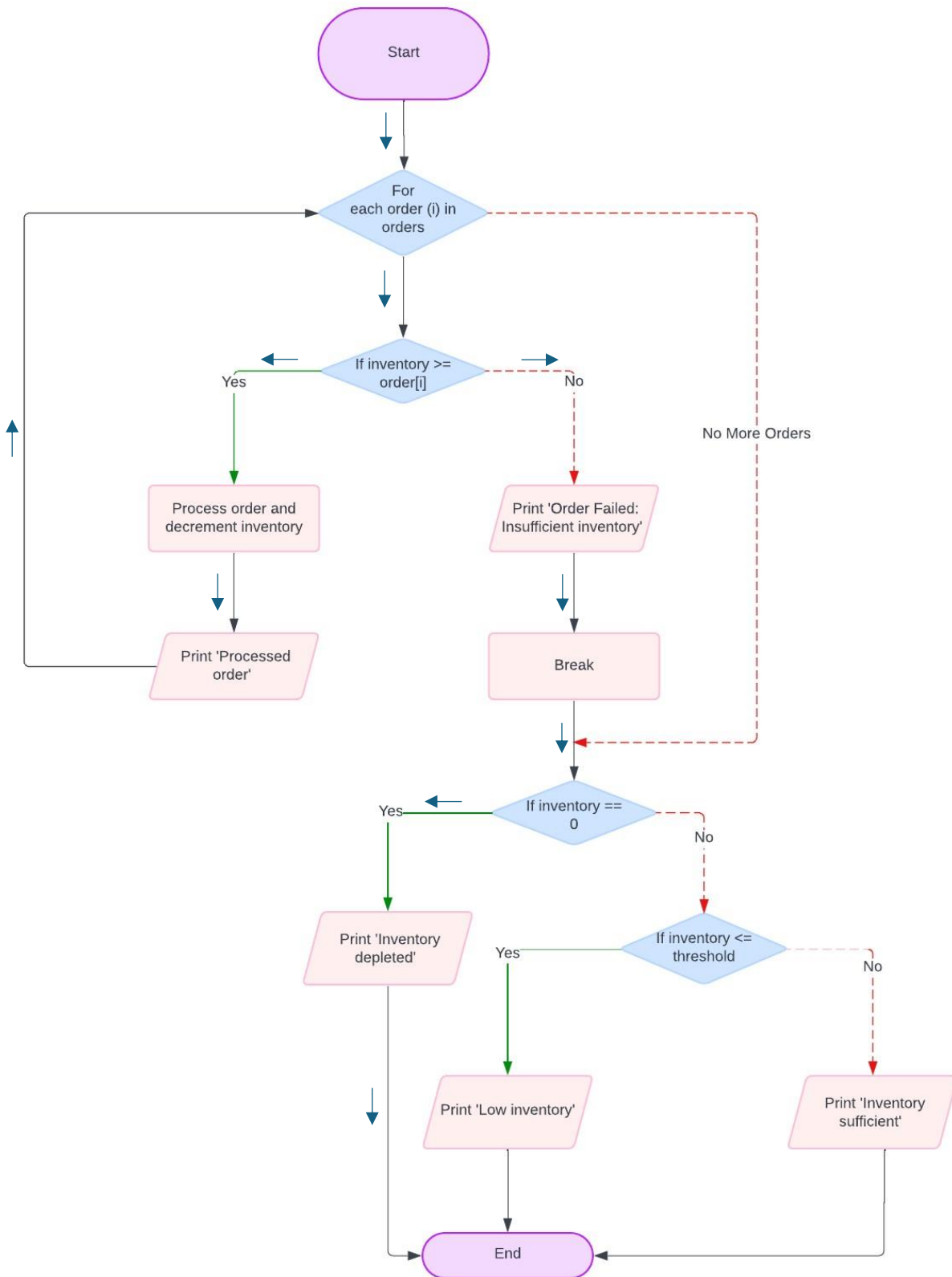
- Độ phủ nhánh C1:

Mã nguồn	Đã thực thi?	TRUE, FALSE?
void manageInventory(int inventory, ...	–	–
{	–	–
for (size_t i = 0; i < orders.size(); ++i) {	O (TRUE, FALSE)	TRUE, FALSE
if (inventory >= orders[i]) {	O (TRUE)	TRUE, FALSE
inventory -= orders[i];	O	–
cout << "Processed order " << i + 1 ...	O	–
} else {	O (FALSE)	–
cout << "Order " << i + 1 << " failed...	O	–
break;	O	–
}	–	–
}	–	–
if (inventory == 0) {	O (TRUE)	TRUE
cout << "Inventory depleted." << endl;	O	–
} else if (inventory <= threshold) {	X (FALSE)	–
cout << "Low inventory: " << inventory ...	X	–
} else {	X (FALSE)	–
cout << "Inventory sufficient: " << ...	X	–
}	–	–
}	–	–

Độ phủ nhánh $C1 = \frac{5}{7} \approx 71.4\%$

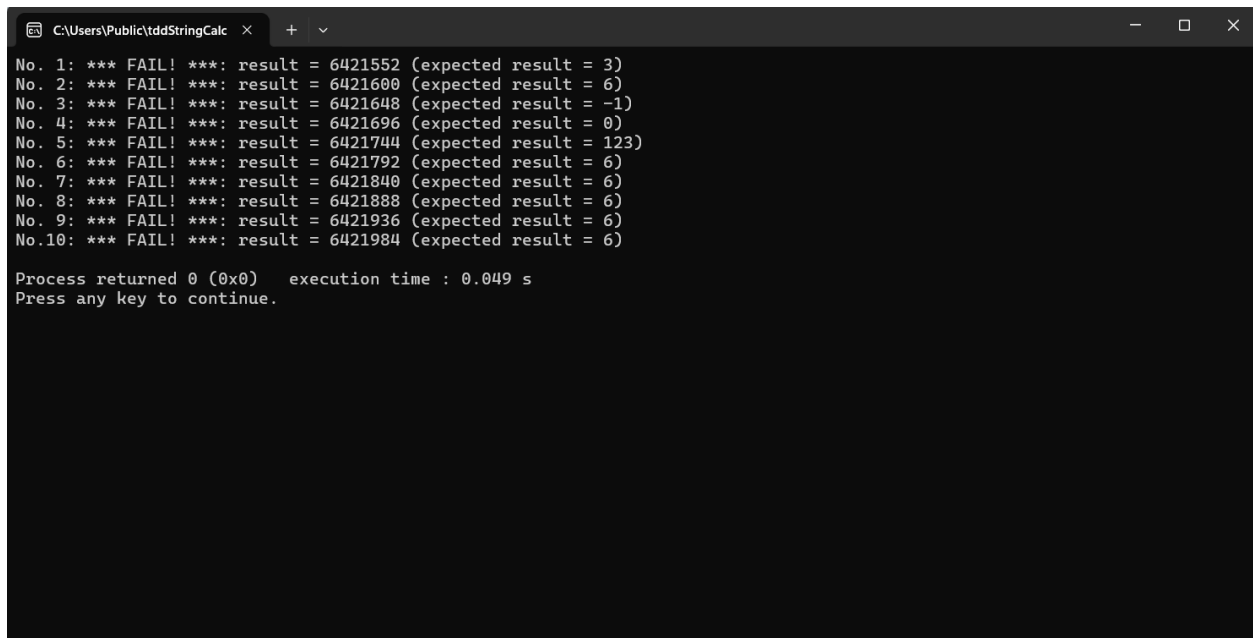
- Độ phủ điều kiện C2:

Độ phủ điều kiện $C2 = \frac{4}{12} \approx 33.3\%$



Bài tập 3.1. Phát triển hướng kiểm thử

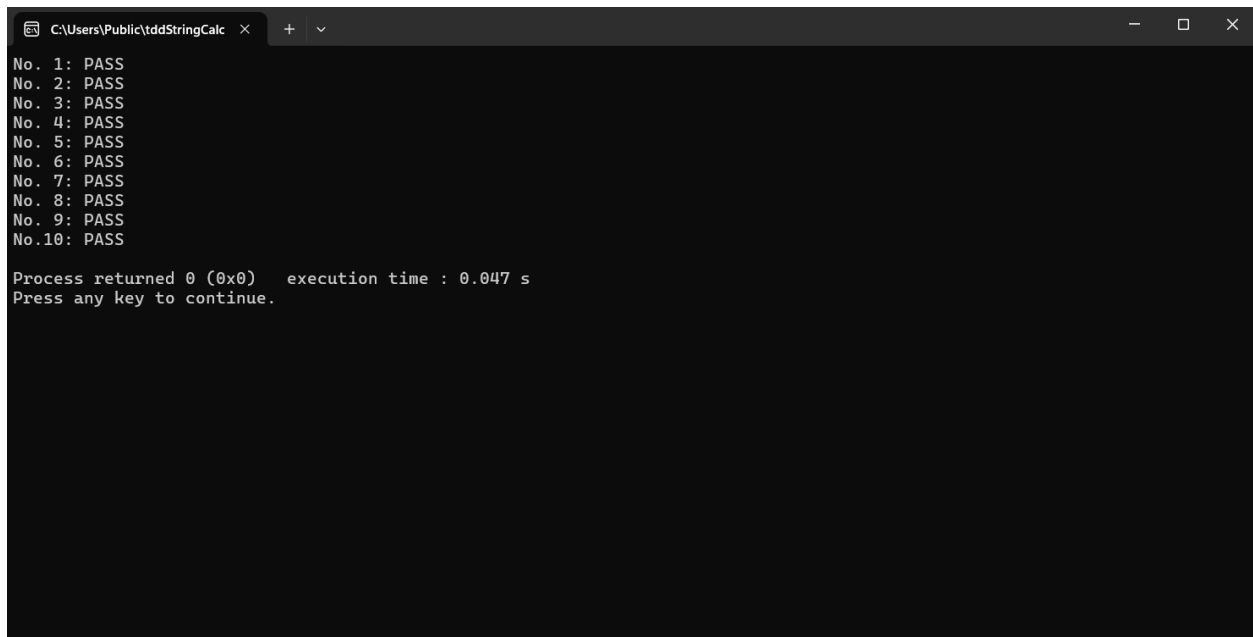
Hình ảnh kết quả kiểm thử tự động lần đầu:



```
C:\Users\Public\tddStringCalc x + v
No. 1: *** FAIL! ***: result = 6421552 (expected result = 3)
No. 2: *** FAIL! ***: result = 6421600 (expected result = 6)
No. 3: *** FAIL! ***: result = 6421648 (expected result = -1)
No. 4: *** FAIL! ***: result = 6421696 (expected result = 0)
No. 5: *** FAIL! ***: result = 6421744 (expected result = 123)
No. 6: *** FAIL! ***: result = 6421792 (expected result = 6)
No. 7: *** FAIL! ***: result = 6421840 (expected result = 6)
No. 8: *** FAIL! ***: result = 6421888 (expected result = 6)
No. 9: *** FAIL! ***: result = 6421936 (expected result = 6)
No.10: *** FAIL! ***: result = 6421984 (expected result = 6)

Process returned 0 (0x0)   execution time : 0.049 s
Press any key to continue.
```

Hình ảnh kết quả kiểm thử tự động sau khi chỉnh sửa chương trình:



```
C:\Users\Public\tddStringCalc x + v
No. 1: PASS
No. 2: PASS
No. 3: PASS
No. 4: PASS
No. 5: PASS
No. 6: PASS
No. 7: PASS
No. 8: PASS
No. 9: PASS
No.10: PASS

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

```
PS C:\Users\Public\Documents\IE108> npx jest --coverage
PASS ./price.calculator.test.js
  calculatePrice
    ✓ should calculate the price correctly without discounts or delivery charges (8 ms)
    ✓ should apply 10% price reduction if price reaches 200 euros (1 ms)
    ✓ should apply delivery charges if weight exceeds 5 kg (1 ms)
    ✓ should apply free delivery charges if price exceeds 100 euros
    ✓ should apply 3% price reduction in reduced price if paid with credit card (1 ms)
    ✓ should apply 15% price reduction if price reaches 200 euros,paid with credit card, and weight is under 5 kg (1 ms)
    ✓ should apply 10% price reduction and an additional 3% reduction if price reaches 200 euros,paid with credit card, and weight is 5 kg or more (1 ms)

-----
File                                     % Stmts   % Branch   % Funcs   % Lines   Uncovered Line #s
-----
All files                               100        100        100        100
price.calculator.js                    100        100        100        100
-----

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        0.72 s, estimated 1 s
Ran all test suites.
PS C:\Users\Public\Documents\IE108>
```