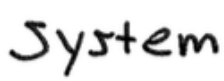
Short-Answer Questions:

1. In _____ testing, you hook everything together and treat the system like a black box.

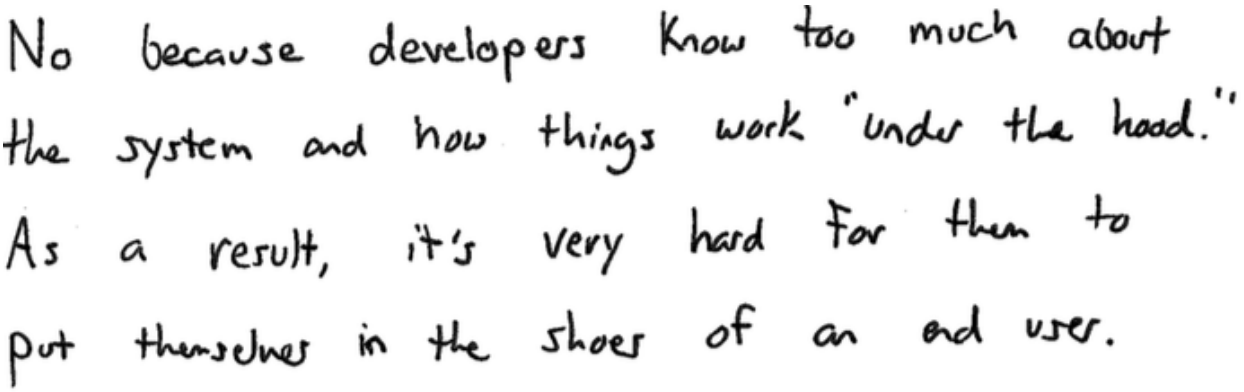
2. When it comes to the type of testing from the previous question, should the software developers who wrote the code perform the testing? Why?

Solutions:

1.



2.



Problem: Give two reasons why it's bad for developers to system test their own code.

Solution:

Many possible answers...

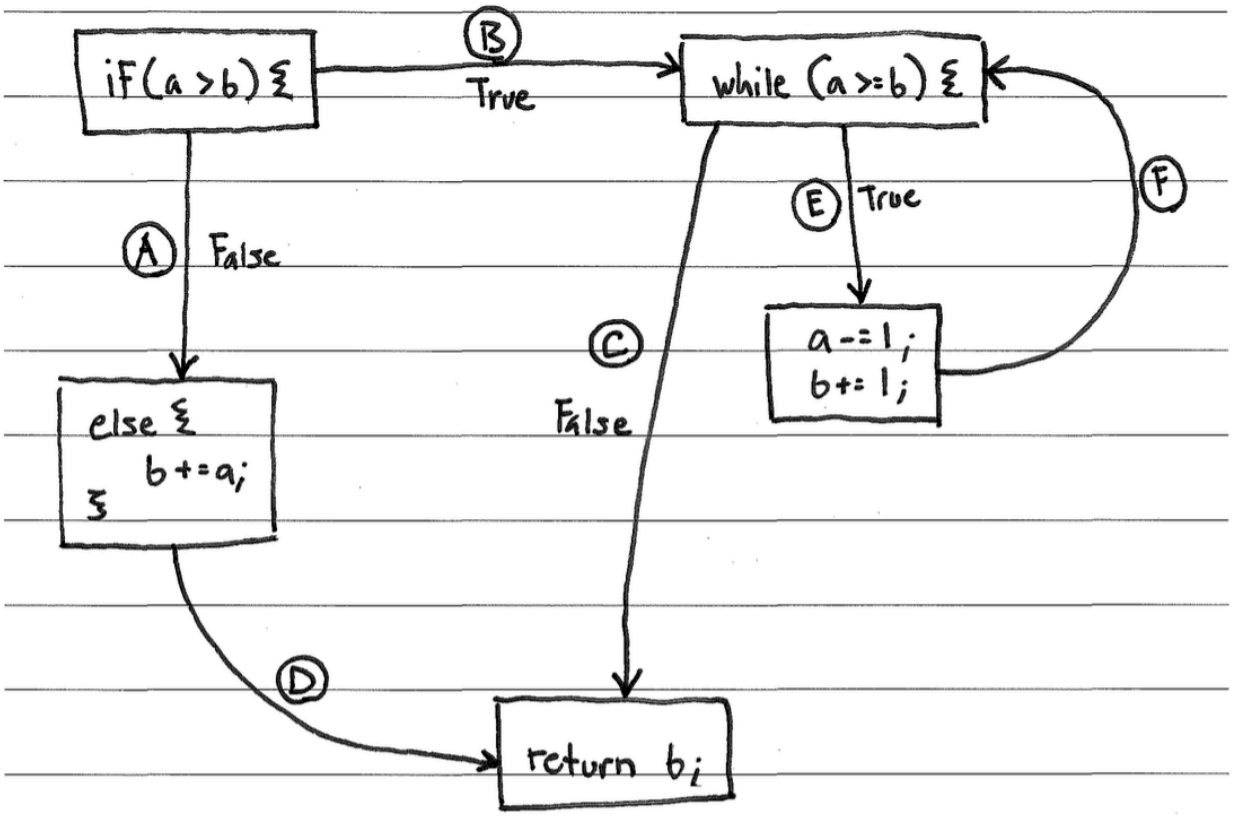
Here [redacted] two good ones:

- Developer can't see system like a user does (he/she knows too much about its inner workings)
- Because developer made it, he/she may have disincentive to find problems

Problem: Draw a control flow diagram for this function. Label each edge with an uppercase letter.

```
int funWithNumbers(int a, int b) {
    if (a > b) {
        while (a >= b) {
            a -= 1;
            b += 1;
        }
    } else {
        b += a;
    }
    return b;
}
```

Solution:


iF(a >b) {
B
True
while (a >=b) {
A
False
E
True
F
a -=1;
b+= 1;
C
False
else {
b+=a;
}
D
return b;

Problem: Fill in the table below with a test suite that provides *path coverage* of the code from the previous question. Cover no more than 2 iterations of the loop. In the covers column, list the relevant labeled items in your CFG that each test case covers. If there is some part of the coverage that is impossible to cover, then list it in the covers column, and put "N/A" in the associated x and y cells. Some cells in the table may be left blank.

Input		**Covers**
x	**y**	

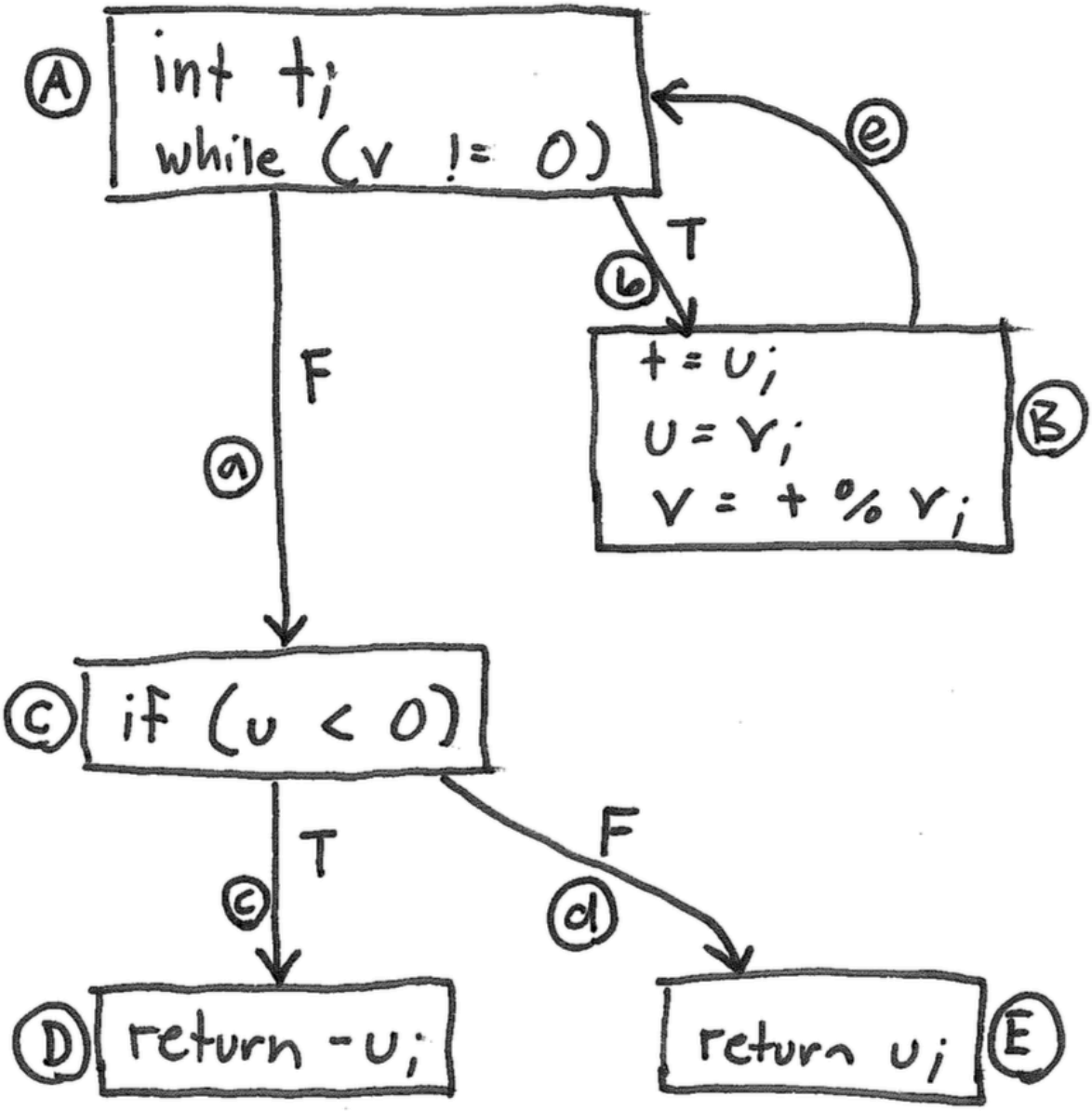
Solution:

Input		Covers
x	y	
1	2	AD
N/A	N/A	BC
1	0	BEFC
4	2	BEFEFC

Problem: Draw a control flow diagram for this function. Label each node in the graph with a capital letter, and label each edge with a lowercase letter.

```
int blammo(int u, int v) {
  int t;
  while (v != 0) {
    t = u;
    u = v;
    v = t % v; // Recall that % computes remainder of t/v
  }
  if (u < 0) { return -u; }
  return u;
}
```

Solution:


A
int t;
while (v != 0)
e
T
b
t = u;
u = v;
v = t % v;
B
F
a
c
if (u < 0)
T
F
c
d
D
return -u;
return u;
E

Problems:

1. Fill in the table below with a test suite that provides statement coverage of the "blammo" code. In the covers column, list the relevant labeled items in your CFG that each test case covers. Some cells in the table may be left blank.

Input		Covers
u	v	

2. Fill in the table below with a test suite that provides path coverage of the "blammo" code. Cover no more than 1 iteration of the loop. In the covers column, list the relevant labeled items in your CFG that each test case covers. Some cells in the table may be left blank.

Input		Covers
u	v	

Solutions:

1.

Input		Covers
u	v	
2	2	A, B, C, E
-1	0	A, C, D

2.

Input		Covers
u	v	
-1	0	a, c
0	0	a, d
-2	-2	b, e, a, c
2	2	b, e, a, d

Paths:

a, c
a, d
b, e, a, c
b, e, a, d