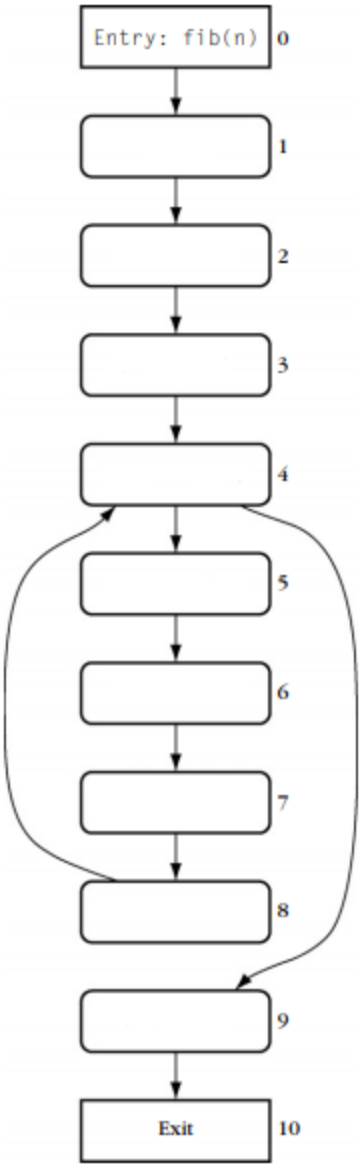
```c
#include <stdio.h>

int fib(int n);

int main(void) {
    int n = 9;
    while (n > 0) {
        printf("fib(%d) = %d", n, fib(n));
        n = n - 1;
    }
    return 0;
}

// statement coverage = number of executed statments / total number of statements

int fib(int n) {
    int f;
    int f0 = 1;
    int f1 = 1;
    while (n > 1) {
        n = n  - 1;
        f = f0 + f1;
        f0 = f1;
        f1 = f;
    }
    return f;
}
```

For the following program fib(1) comes to a problem. You are experienced programmers and I have full confidence that you can identify why quickly. But let us use more systematic means to find out why. The first thing to reason about when tracking value origins through source code is to identify those regions of code that could have influenced the value *simply because they were executed*. In our example, this is particularly easy. We only need to consider the code of the fib() function, as the defect occurs between its call and its return Please complete the control flow graph for each statement below.



$A \longrightarrow B$

Control flow:
A precedes *B*;
B follows *A*