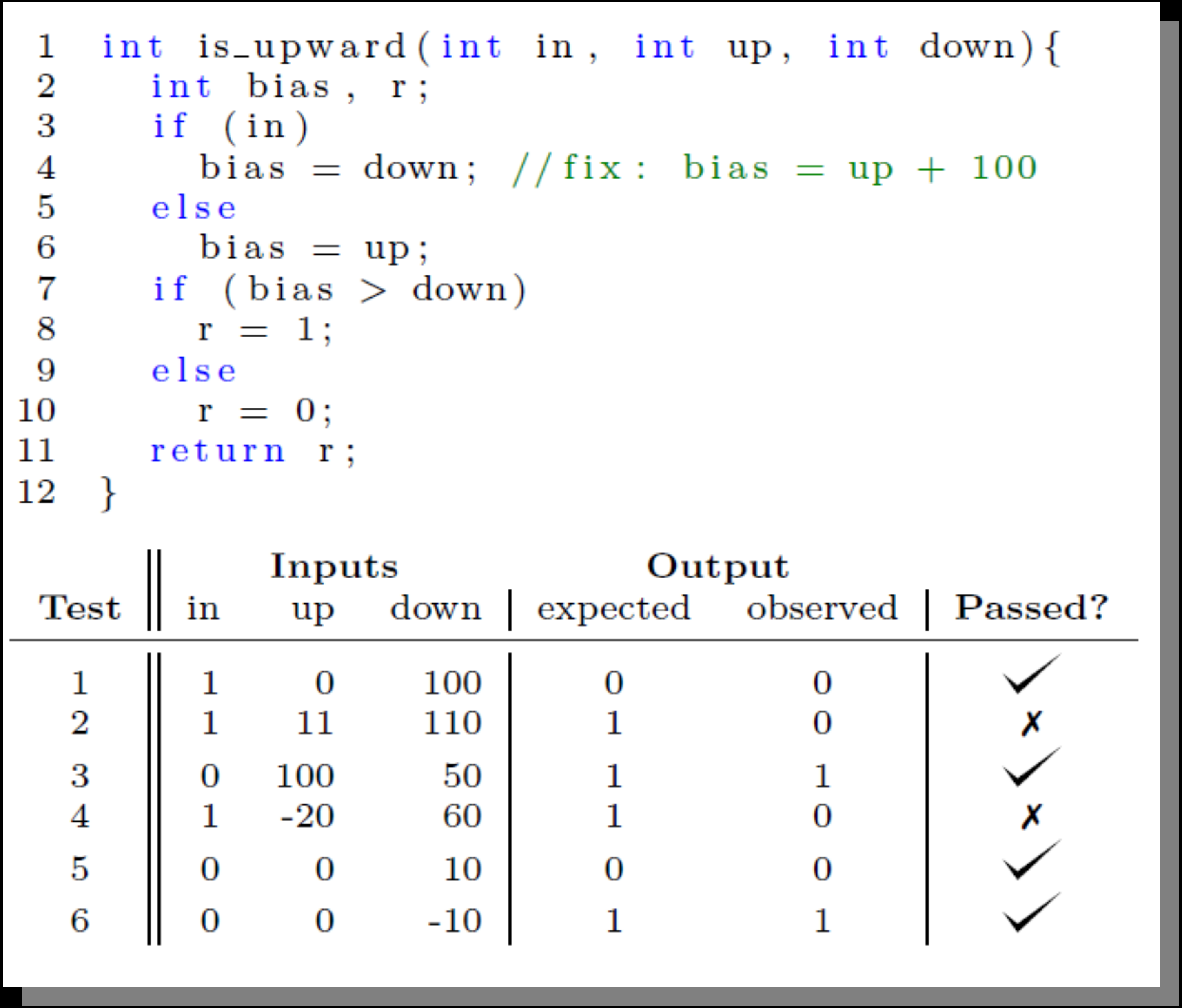
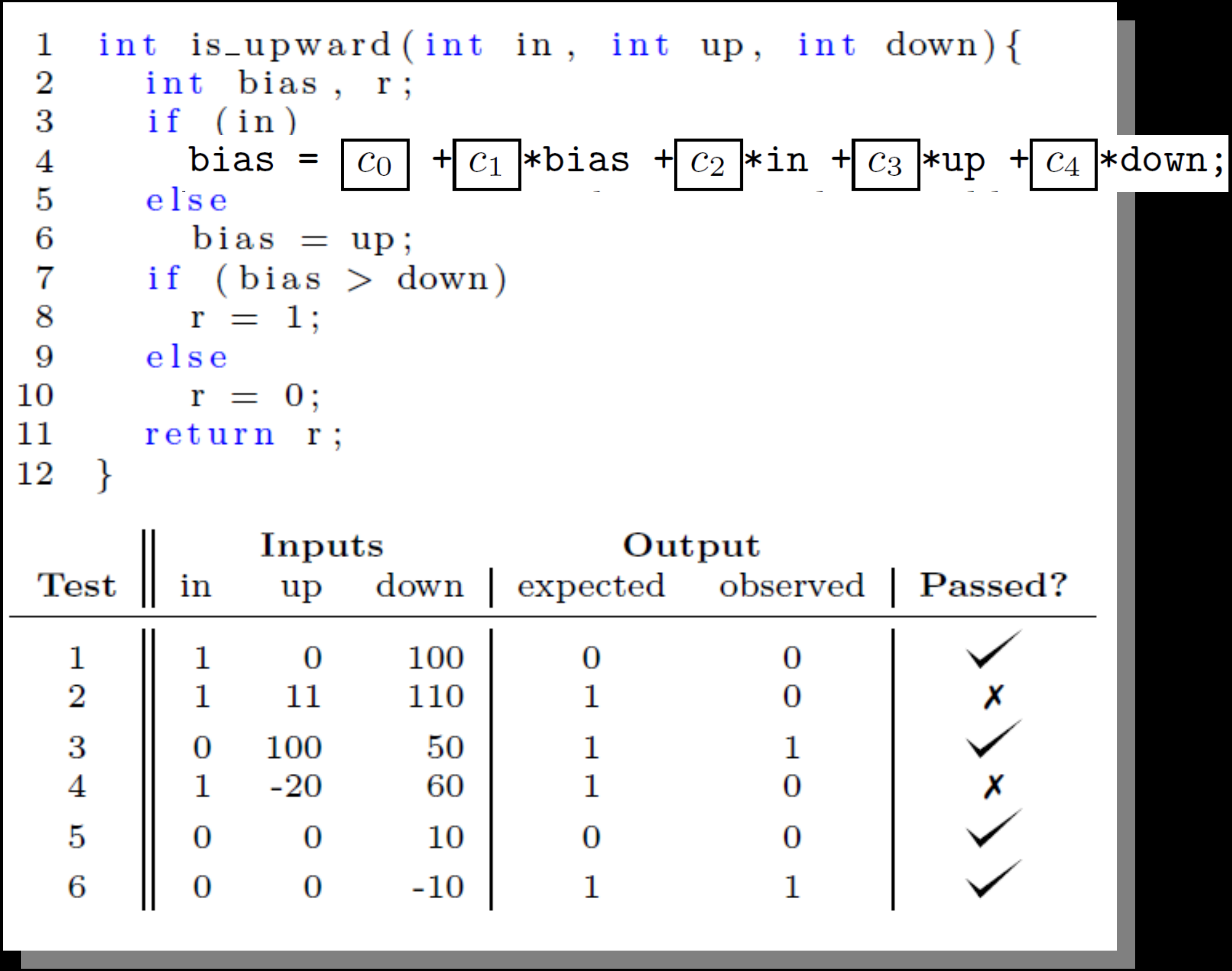
Program Repair Example



```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = down; //fix: bias = up + 100
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

	Inputs			Output		
Test	in	up	down	expected	observed	Passed?
1	1	0	100	0	0	✓
2	1	11	110	1	0	✗
3	0	100	50	1	1	✓
4	1	-20	60	1	0	✗
5	0	0	10	0	0	✓
6	0	0	-10	1	1	✓

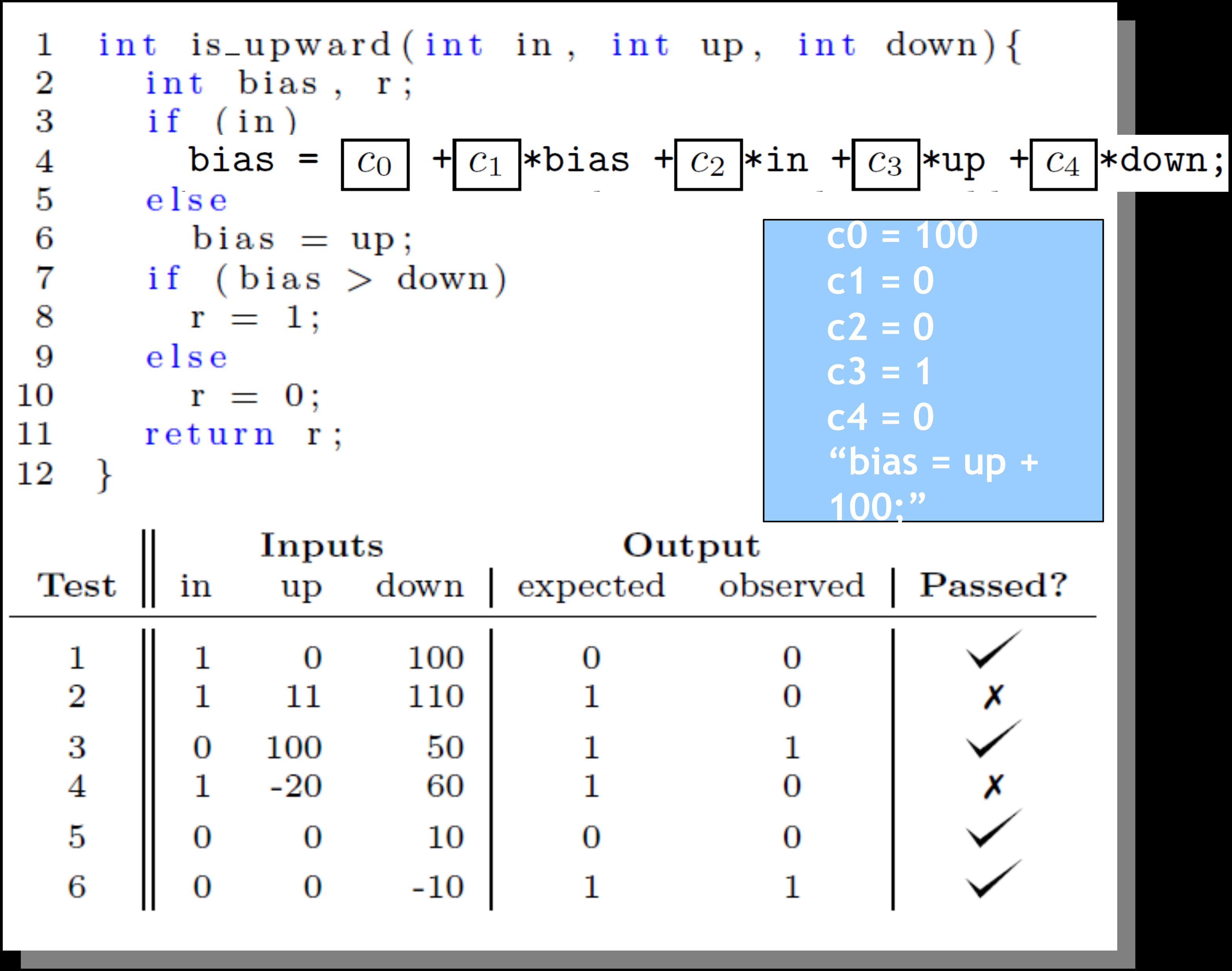
Program Repair Example



```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0 + c1*bias + c2*in + c3*up + c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

	Inputs			Output		
Test	in	up	down	expected	observed	Passed?
1	1	0	100	0	0	✓
2	1	11	110	1	0	✗
3	0	100	50	1	1	✓
4	1	-20	60	1	0	✗
5	0	0	10	0	0	✓
6	0	0	-10	1	1	✓

Program Repair Example

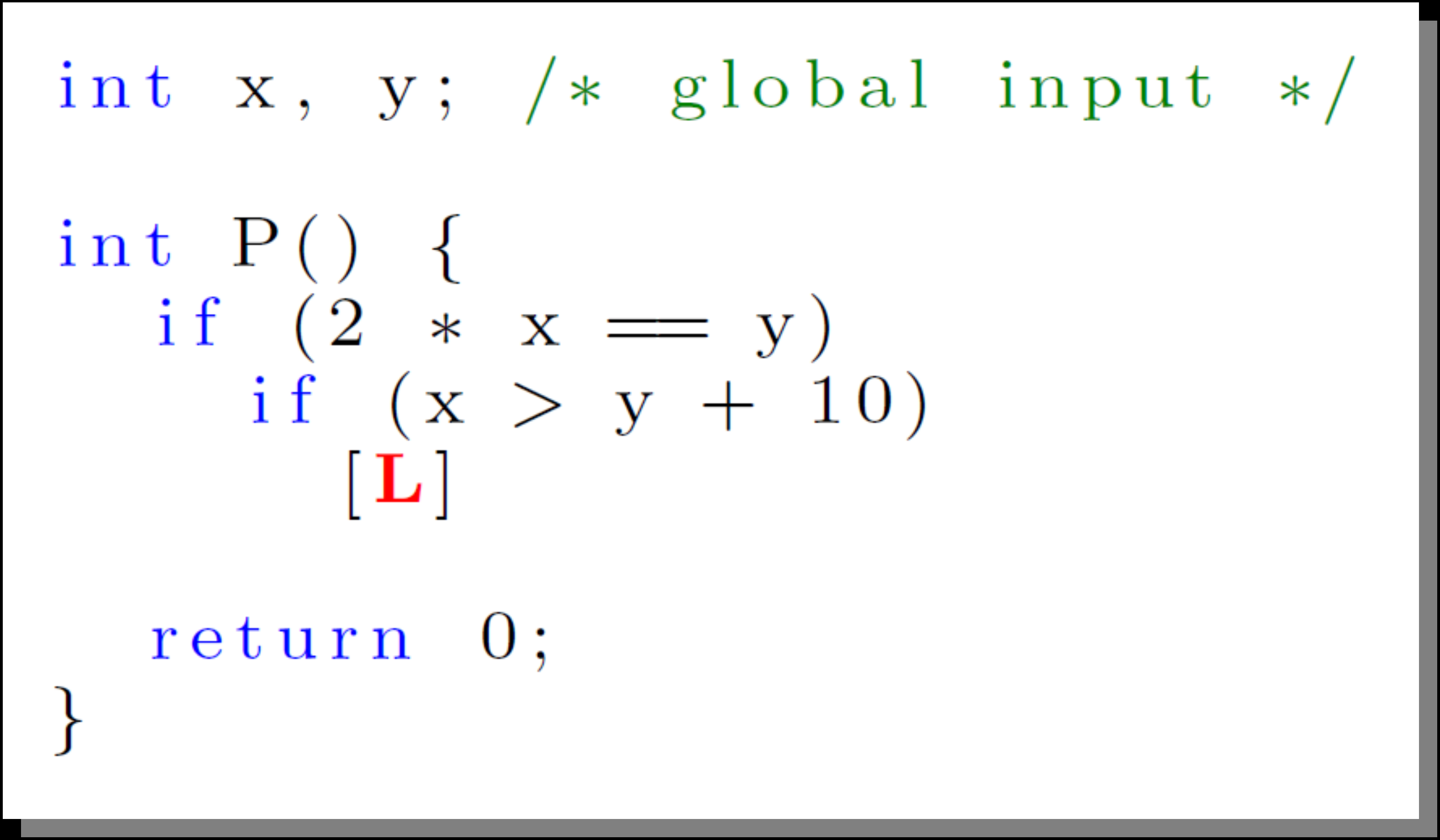


```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0 +c1*bias +c2*in +c3*up +c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

c0 = 100
c1 = 0
c2 = 0
c3 = 1
c4 = 0
“bias = up + 100;”

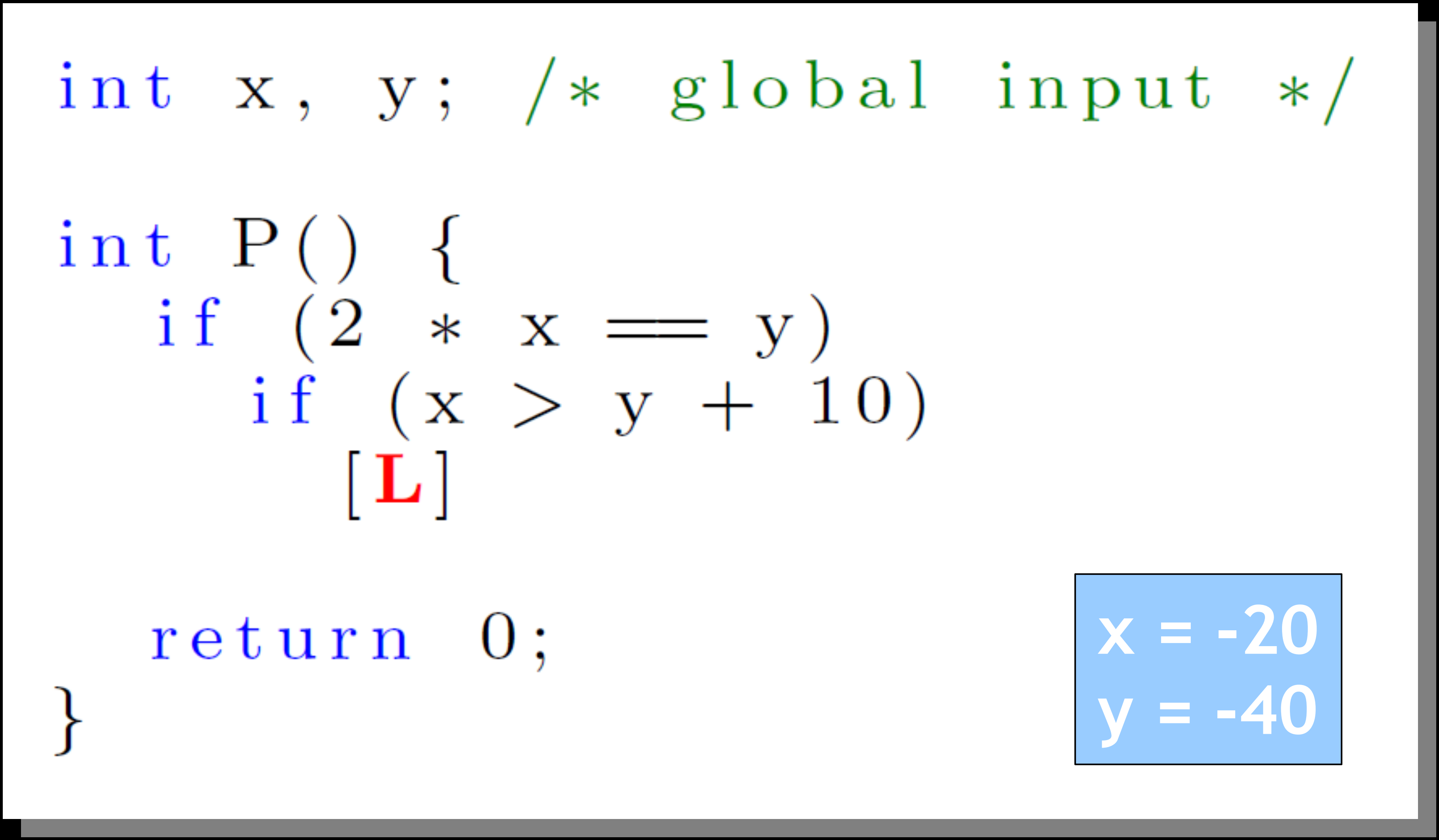
Test	Inputs			Output		Passed?
	in	up	down	expected	observed	
1	1	0	100	0	0	✓
2	1	11	110	1	0	✗
3	0	100	50	1	1	✓
4	1	-20	60	1	0	✗
5	0	0	10	0	0	✓
6	0	0	-10	1	1	✓

Reachability Example



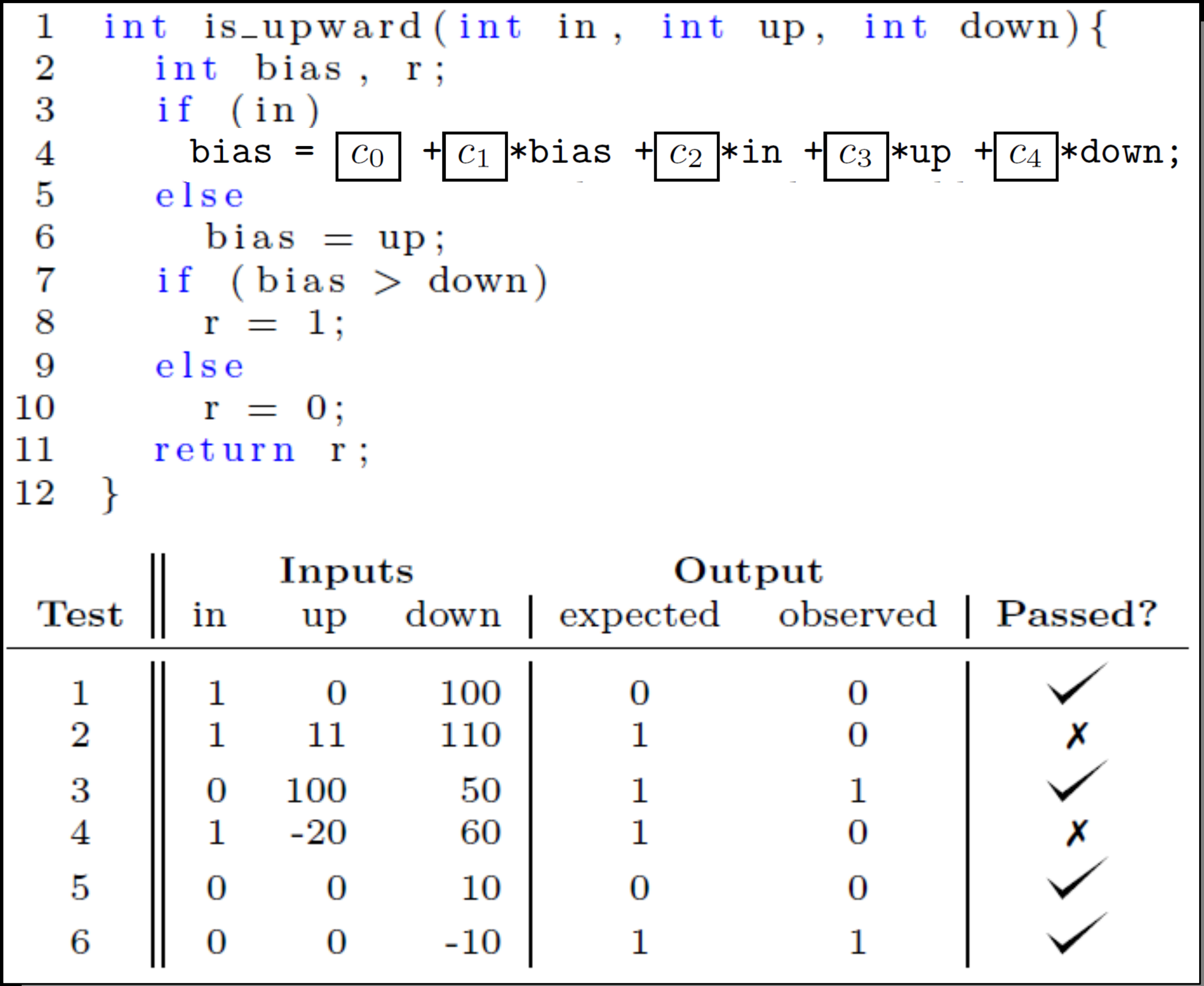
```
int x, y; /* global input */

int P() {
  if (2 * x == y)
    if (x > y + 10)
      [L]

  return 0;
}
```

Reachability Example



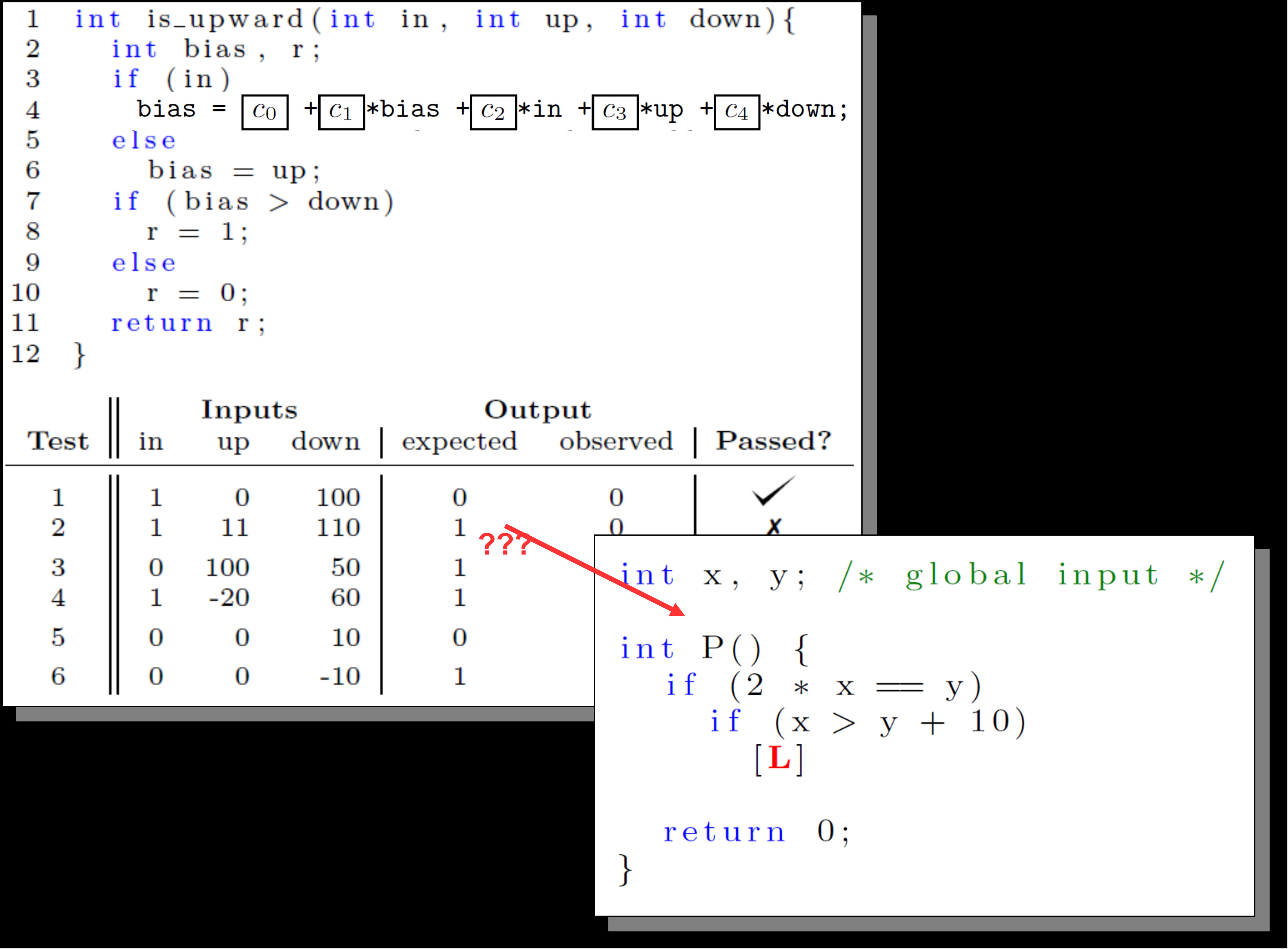
```
int x, y; /* global input */

int P() {
   if (2 * x == y)
      if (x > y + 10)
         [L]

   return 0;
}
```

x = -20
y = -40



```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0 + c1*bias + c2*in + c3*up + c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

Test	Inputs			Output		Passed?
	in	up	down	expected	observed	
1	1	0	100	0	0	✓
2	1	11	110	1	0	✗
3	0	100	50	1	1	✓
4	1	-20	60	1	0	✗
5	0	0	10	0	0	✓
6	0	0	-10	1	1	✓

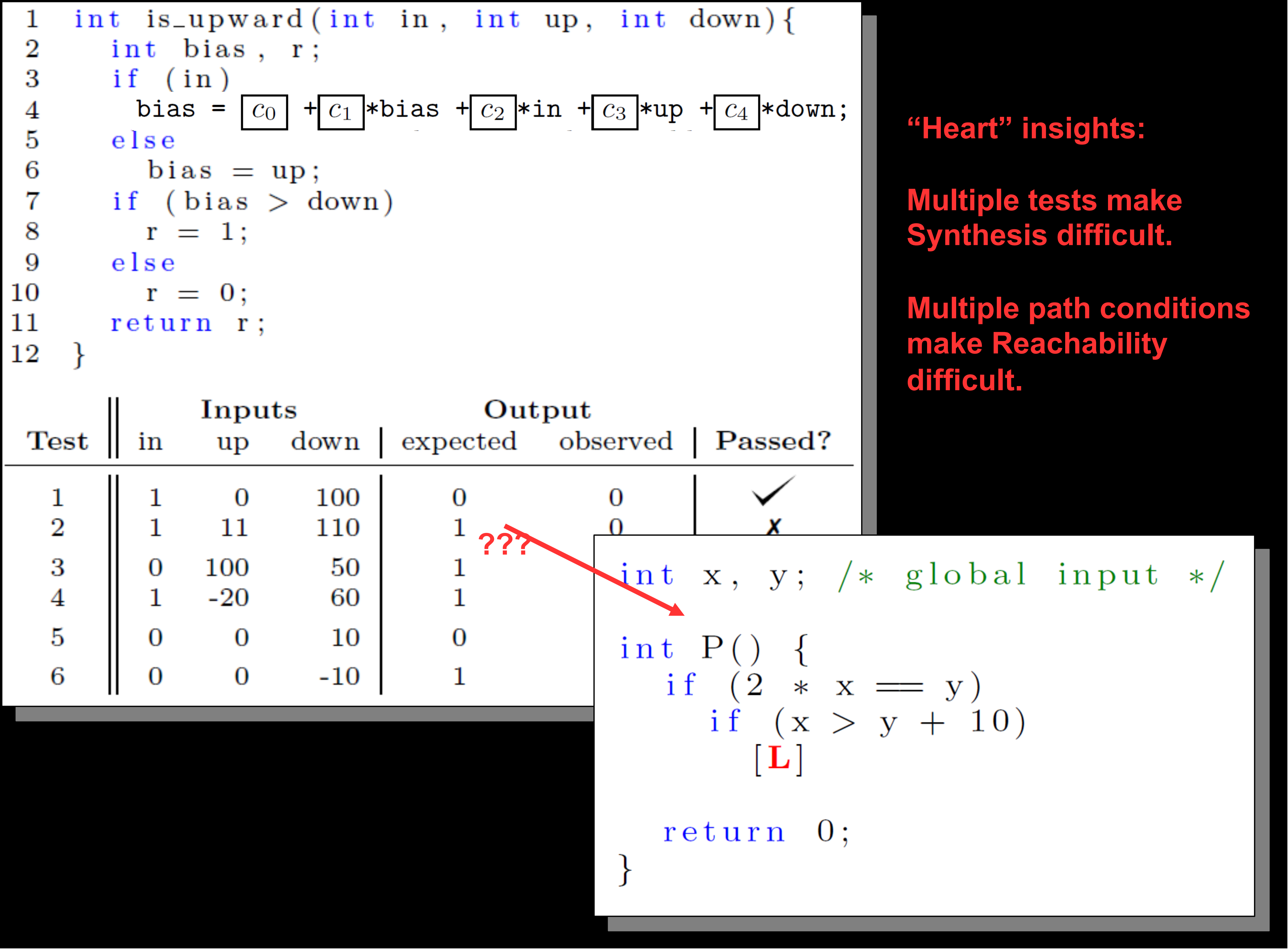


```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0 + c1*bias + c2*in + c3*up + c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

Test	Inputs			Output		Passed?
	in	up	down	expected	observed	
1	1	0	100	0	0	✓
2	1	11	110	1	0	✗
3	0	100	50	1		
4	1	-20	60	1		
5	0	0	10	0		
6	0	0	-10	1		

???

```
int x, y; /* global input */

int P() {
  if (2 * x == y)
    if (x > y + 10)
      [L]

  return 0;
}
```



```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = $\boxed{c_0}$ +$\boxed{c_1}$*bias +$\boxed{c_2}$*in +$\boxed{c_3}$*up +$\boxed{c_4}$*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

	Inputs			Output		
Test	in	up	down	expected	observed	Passed?
1	1	0	100	0	0	✓
2	1	11	110	1	0	✗
3	0	100	50	1		
4	1	-20	60	1		
5	0	0	10	0		
6	0	0	-10	1		

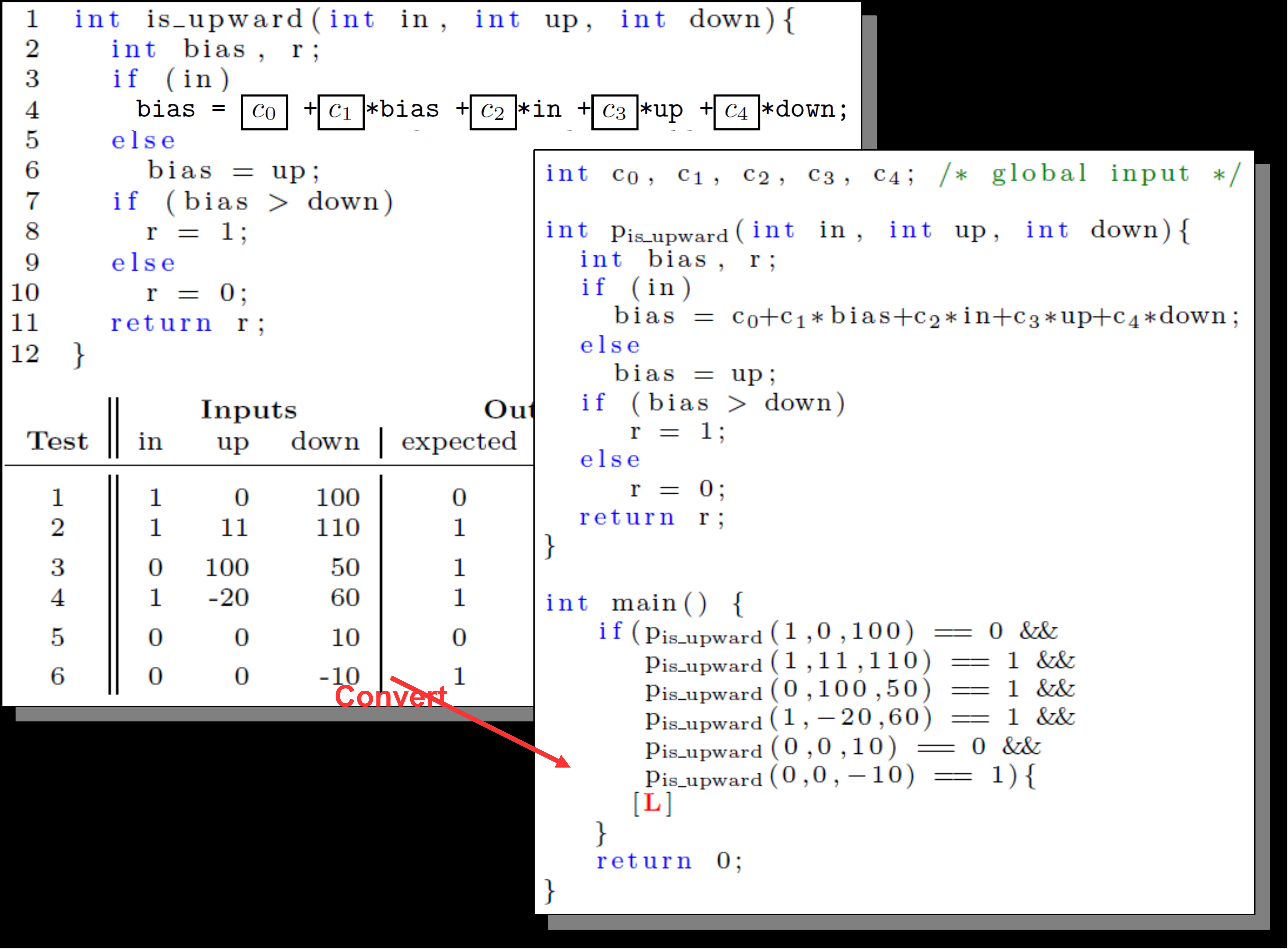
???

```
int x, y; /* global input */

int P() {
  if (2 * x == y)
    if (x > y + 10)
      [L]

  return 0;
}
```

“Heart” insights:

Multiple tests make Synthesis difficult.

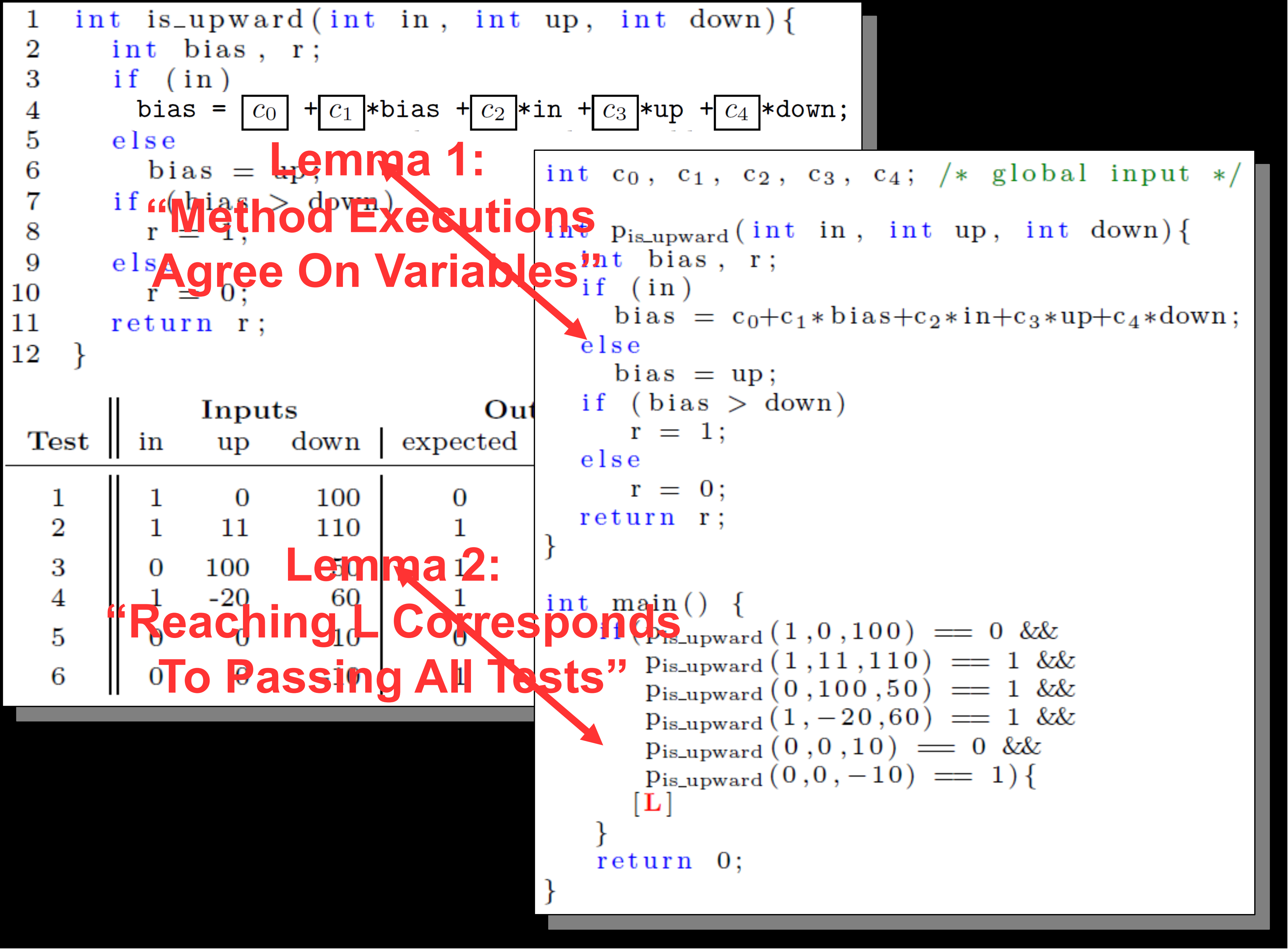
Multiple path conditions make Reachability difficult.



```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0 +c1*bias +c2*in +c3*up +c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

Test	Inputs			Out
	in	up	down	expected
1	1	0	100	0
2	1	11	110	1
3	0	100	50	1
4	1	-20	60	1
5	0	0	10	0
6	0	0	-10	1

```
int c0, c1, c2, c3, c4; /* global input */

int p_is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0+c1*bias+c2*in+c3*up+c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}

int main() {
    if(p_is_upward(1,0,100) == 0 &&
       p_is_upward(1,11,110) == 1 &&
       p_is_upward(0,100,50) == 1 &&
       p_is_upward(1,-20,60) == 1 &&
       p_is_upward(0,0,10) == 0 &&
       p_is_upward(0,0,-10) == 1){
      [L]
    }
    return 0;
}
```



```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0 +c1*bias +c2*in +c3*up +c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

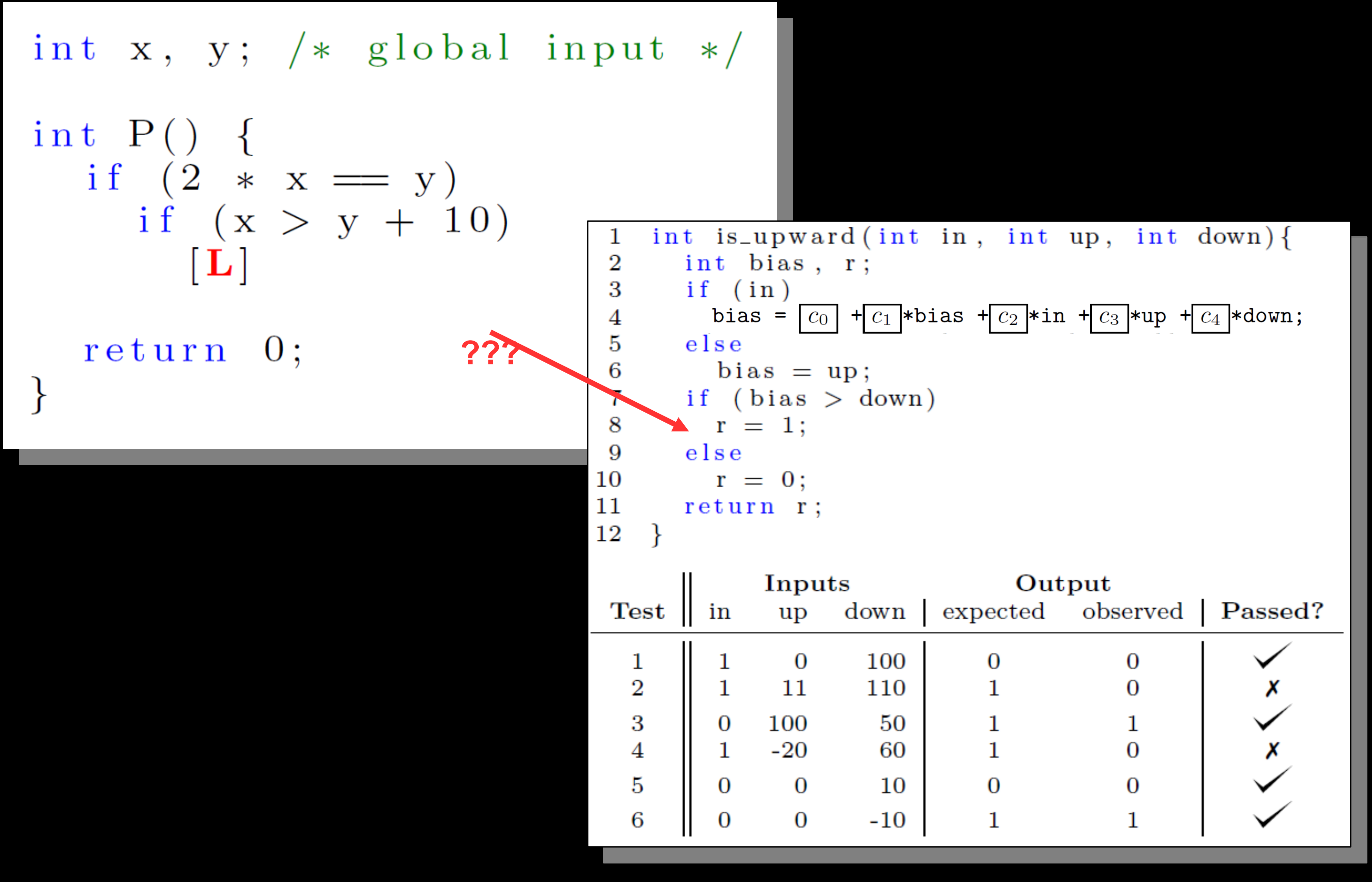
Test	Inputs			Out
	in	up	down	expected
1	1	0	100	0
2	1	11	110	1
3	0	100	50	1
4	1	-20	60	1
5	0	0	10	0
6	0	0	-10	1

```
int c0, c1, c2, c3, c4; /* global input */

int p_is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0+c1*bias+c2*in+c3*up+c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}

int main() {
  if (p_is_upward(1,0,100) == 0 &&
      p_is_upward(1,11,110) == 1 &&
      p_is_upward(0,100,50) == 1 &&
      p_is_upward(1,-20,60) == 1 &&
      p_is_upward(0,0,10) == 0 &&
      p_is_upward(0,0,-10) == 1){
    [L]
  }
  return 0;
}
```

Lemma 1:
“Method Executions Agree On Variables”

Lemma 2:
“Reaching L Corresponds To Passing All Tests”

Reachability to Synthesis Example



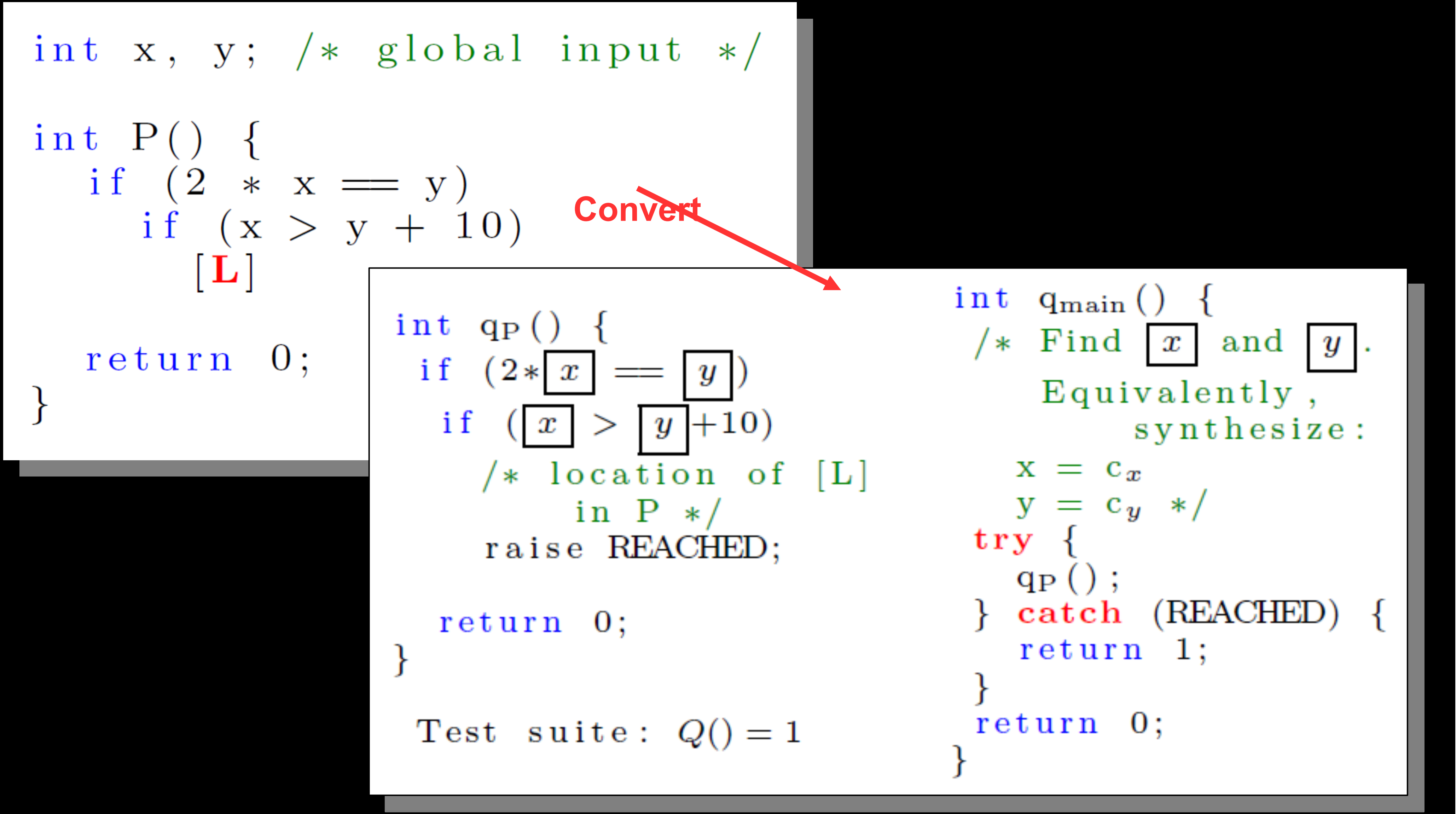
```
int x, y; /* global input */

int P() {
  if (2 * x == y)
    if (x > y + 10)
      [L]

  return 0;
}
```

???

```
int is_upward(int in, int up, int down){
  int bias, r;
  if (in)
    bias = c0 + c1*bias + c2*in + c3*up + c4*down;
  else
    bias = up;
  if (bias > down)
    r = 1;
  else
    r = 0;
  return r;
}
```

Test	Inputs			Output		Passed?
	in	up	down	expected	observed	
1	1	0	100	0	0	✓
2	1	11	110	1	0	✗
3	0	100	50	1	1	✓
4	1	-20	60	1	0	✗
5	0	0	10	0	0	✓
6	0	0	-10	1	1	✓

Reachability to Synthesis Example



```
int x, y; /* global input */

int P() {
  if (2 * x == y)
    if (x > y + 10)
      [L]

  return 0;
}
```

Convert

```
int q_P() {
 if (2*[x] == [y])
  if ([x] > [y]+10)
    /* location of [L]
          in P */
    raise REACHED;

  return 0;
}

 Test suite: Q() = 1
```

```
int q_main() {
 /* Find [x] and [y].
    Equivalently,
        synthesize:
     x = c_x
     y = c_y */
 try {
     q_P();
 } catch (REACHED) {
     return 1;
 }
 return 0;
}
```