

Set 7: Function Testing

Functions under tests are these 2 :

- cli_set_idle_timeout(int)
- crypto_auth_hmacsha256_update(const char*, size_t)

1. Function Implementation

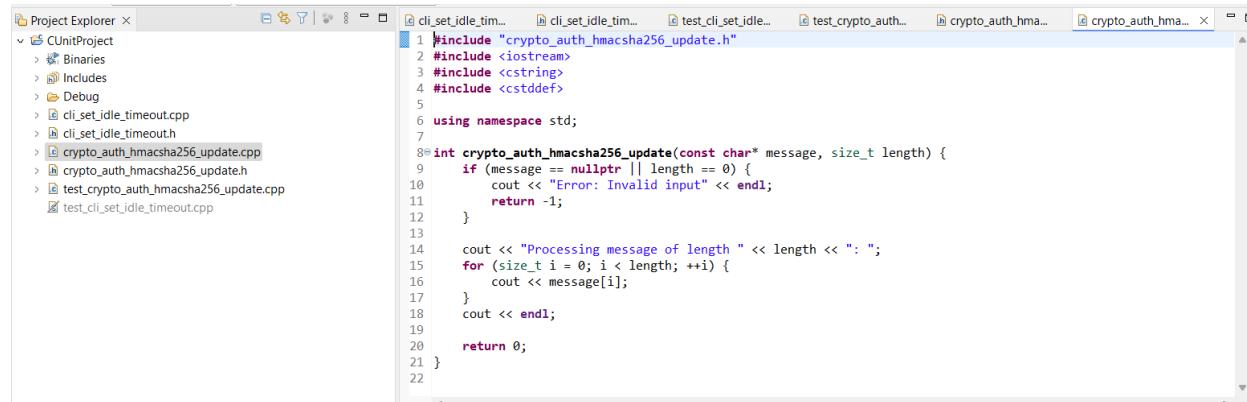
- cli_set_idle_timeout(int)



The screenshot shows a CUnit Project structure in the Project Explorer. The file `cli_set_idle_timeout.cpp` is open, displaying its implementation:

```
1 #include "cli_set_idle_timeout.h"
2
3 int cli_set_idle_timeout(int timeout) {
4     if (timeout < 0) return -1;
5     return 0;
6 }
7
```

- crypto_auth_hmacsha256_update(const char*, size_t)



The screenshot shows a CUnit Project structure in the Project Explorer. The file `crypto_auth_hmacsha256_update.cpp` is open, displaying its implementation:

```
1 #include "crypto_auth_hmacsha256_update.h"
2
3 #include <iostream>
4 #include <cstring>
5 #include <cstddef>
6
7 using namespace std;
8
9 int crypto_auth_hmacsha256_update(const char* message, size_t length) {
10    if (message == nullptr || length == 0) {
11        cout << "Error: Invalid input" << endl;
12        return -1;
13    }
14    cout << "Processing message of length " << length << ": ";
15    for (size_t i = 0; i < length; ++i) {
16        cout << message[i];
17    }
18    cout << endl;
19
20    return 0;
21 }
22
```

2. Test Scripts

- CUnit test cases were written and executed for `cli_set_idle_timeout()`.

The screenshot shows a code editor with multiple tabs open at the top. The active tab is 'test_cli_set_idle_time...' which contains the following C code:

```
1 #include <CUUnit/CUnit.h>
2 #include <CUUnit/Basic.h>
3 #include "cli_set_idle_timeout.h" // ☐ not the .cpp file
4
5 void test_valid_timeout() {
6     CU_ASSERT(cli_set_idle_timeout(10) == 0);
7 }
8
9 void test_invalid_timeout() {
10    CU_ASSERT(cli_set_idle_timeout(-5) == -1);
11 }
12
13 int main() {
14     CU_initialize_registry();
15     CU_pSuite suite = CU_add_suite("IdleTimeoutSuite", 0, 0);
16     CU_add_test(suite, "Test valid timeout", test_valid_timeout);
17     CU_add_test(suite, "Test invalid timeout", test_invalid_timeout);
18     CU_basic_run_tests();
19     CU_cleanup_registry();
20     return 0;
21 }
```

- Manual main-based test calls were created and verified for crypto_auth_hmacsha256_update().

The screenshot shows a code editor with multiple tabs open at the top. The active tab is 'test_crypto_auth_hmac...' which contains the following C code:

```
1 #include "crypto_auth_hmacsha256_update.h"
2 #include "cli_set_idle_timeout.h"
3
4 #include <iostream>
5 #include <cstring>
6 using namespace std;
7
8 int main() {
9     const char* msg = "Test message";
10    crypto_auth_hmacsha256_update(msg, strlen(msg)); // ☐ no error now
11    return 0;
12 }
13 |
```

3. Execution & Screenshot

- CUnit test cases for cli_set_idle_timeout executed successfully

The screenshot shows the Eclipse IDE interface with the CUnit Project Debug perspective selected. The Project Explorer shows files like cli_set_idle_timeout.cpp, cli_set_idle_timeout.h, and test_cli_set_idle_timeout.cpp. The code editor displays the test_cli_set_idle_timeout.cpp file containing CUnit test cases for the cli_set_idle_timeout function. The Problems view shows a terminated build with exit value 0. The Console view displays the test results:

```
CUnit - A unit testing framework for C - Version 2.1-3
http://cunit.sourceforge.net/

Run Summary: Type Total Ran Passed Failed Inactive
suites 1 1 n/a 0 0
tests 2 2 2 0 0
asserts 2 2 2 0 n/a

Elapsed time = 0.000 seconds
```

- Console test output for crypto_auth_hmacsha256_update

The screenshot shows the Eclipse IDE interface with the C/C++ Application perspective selected. The Project Explorer shows files like cli_set_idle_timeout.cpp, cli_set_idle_timeout.h, test_cli_set_idle_timeout.cpp, test_crypto_auth_hmacsha256_update.cpp, and crypto_auth_hmacsha256_update.cpp. The code editor displays the test_crypto_auth_hmacsha256_update.cpp file containing a main function that calls the crypto_auth_hmacsha256_update function with a test message. The Problems view shows a terminated build with exit value 0. The Console view displays the test results:

```
<terminated> (exit value: 0) CUnitProject Debug [C/C++ Application] C:\Users\DELL\cpp-eclipse-workspace\CUnitProject\Debug\CUnitProject.exe (19/4/25, 11:00:00)

Processing message of length 12: Test message
```

2. Input Space Partitioning For Both Functions

Function 1: cli_set_idle_timeout(int timeout)

Input Parameter:

This function takes a single input:

int timeout

Input Characteristic Identified:

- **Timeout Value Type** — the type of value being passed (negative, zero, positive)

Input Characteristic	Partition	Test Value
Timeout Value	Negative	-5
	Zero	0
	Positive	10

- Any **negative value** should return -1 indicating invalid input.
 - Any **zero or positive value** should return 0 indicating valid input.
-

Function 2: `crypto_auth_hmacsha256_update(const char* message, size_t length)`

Two input parameters:

1. `message` → a pointer to a C-style string
2. `length` → length of the message

Identify Input Characteristics:

Characteristic 1: Message Validity

- Is the pointer null?
- Is it an empty string?
- Is it a valid non-empty string?

Characteristic 2: Length Validity

- Is the length 0?
- Is the length matching the string size?
- Is it longer/shorter than actual string?

Input Characteristic	Partition	Test Value
Message	Null Pointer	nullptr
	Empty String	""
	Valid String	"Test"
Length	Zero Length	0
	Correct Length	4 (for "Test")
	Too Long (Edge Case)	10 (for "Test")

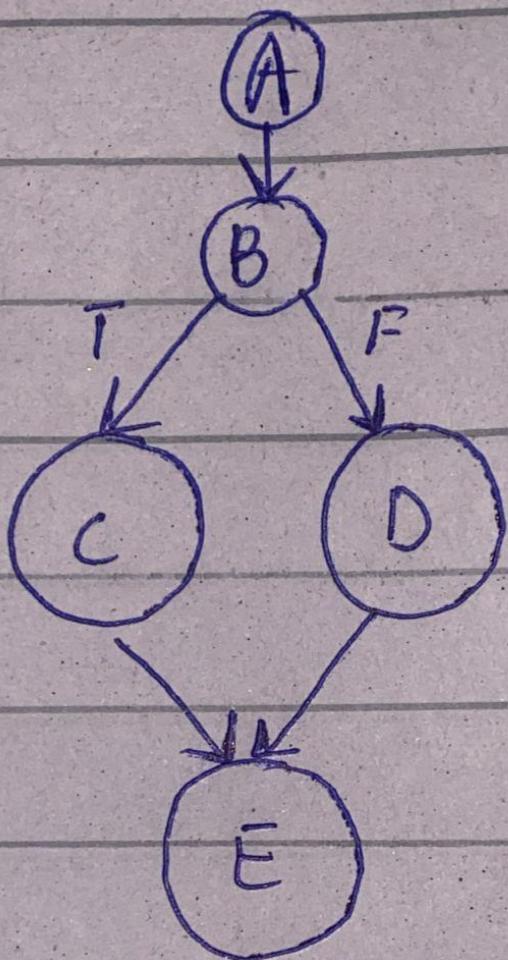
3. Edge-Pair Coverage (EPC) + Control Flow Graph (CFG)

Function 1: cli_set_idle_timeout(int timeout)

Code for CFG:

```
int cli_set_idle_timeout(int timeout) {
    if (timeout < 0)
        return -1;
    return 0;
}
```

Control Flow Graph (CFG):



Edge-Pairs:

Edge-Pair	Path
A → B → C	Negative timeout
A → B → D	Zero or positive

EPC Test Values:

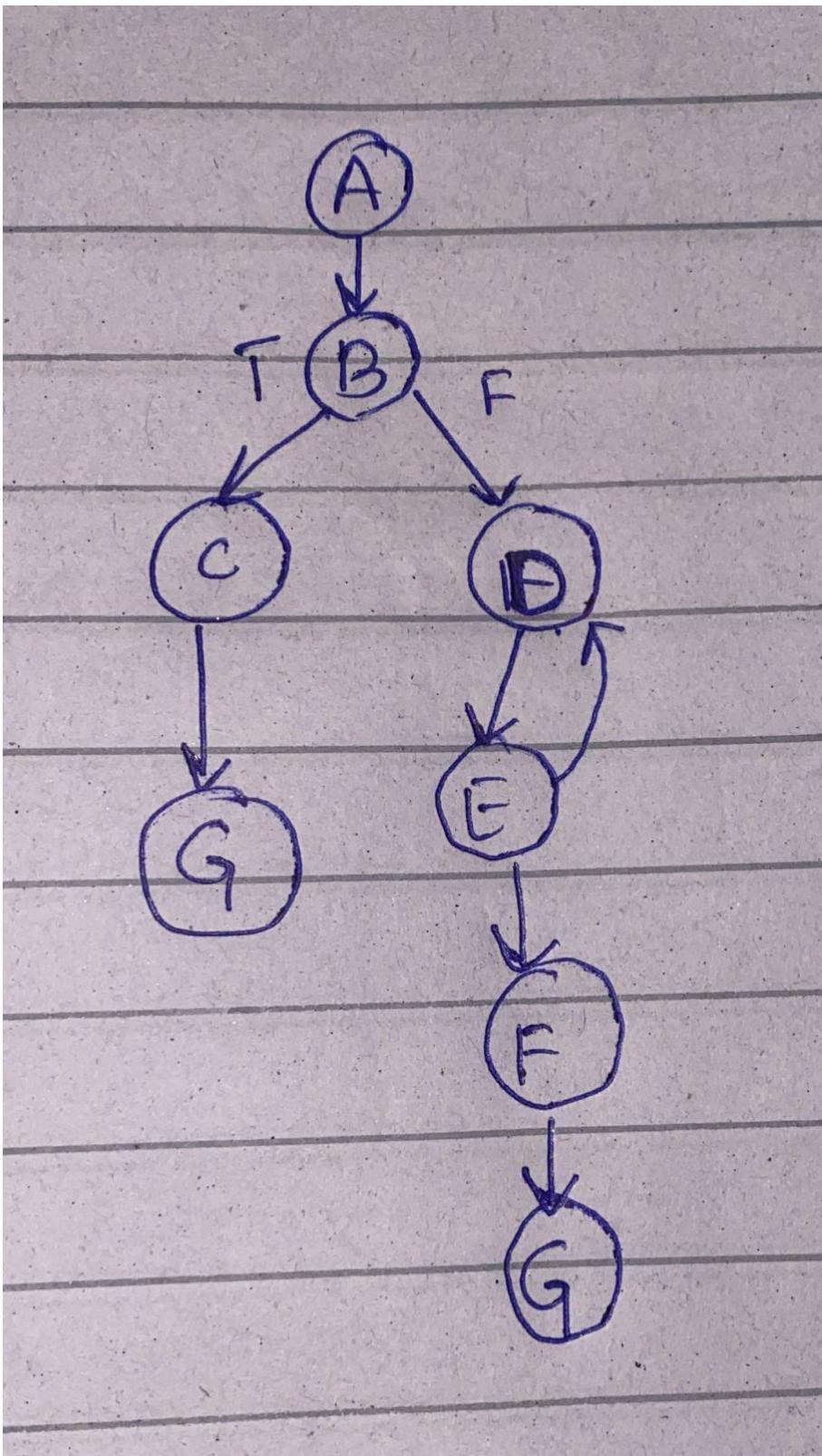
Edge-Pair	Test Value	Explanation
A → B → C → E	-5	Negative → returns -1
A → B → D → E	10, 0	Valid → returns 0

Function 2: crypto_auth_hmacsha256_update(const char* message, size_t length)

Code for CFG:

```
int crypto_auth_hmacsha256_update(const char* message, size_t length) {
    if (message == nullptr || length == 0) {
        return -1;
    }
    // Print message
    for (size_t i = 0; i < length; ++i) {
        cout << message[i];
    }
    return 0;
}
```

Control Flow Graph (CFG):



Edge Pairs:

Edge-Pair	Description
A → B → C	Message is null or length is 0
A → B → D	Message valid, length > 0
D → E → D	Loop executes ($i < \text{length}$)
D → F	Loop exit when $i \geq \text{length}$
C → G	Return -1 on invalid input
F → G	Return 0 after printing

EPC Test Values:

Edge Pair Path	Input Values	Expected Output
A → B → C → G	nullptr, 5	-1 (invalid input)
A → B → C → G	"Hello", 0	-1 (invalid input)
A → B → D → F → G	"Hi", 2 → loop runs twice	0 + prints Hi
D → E → D (loop)	"Hi", 2 → tests loop condition	Intermediate printing
D → F → G	After $i == \text{length}$	Ends printing, return 0