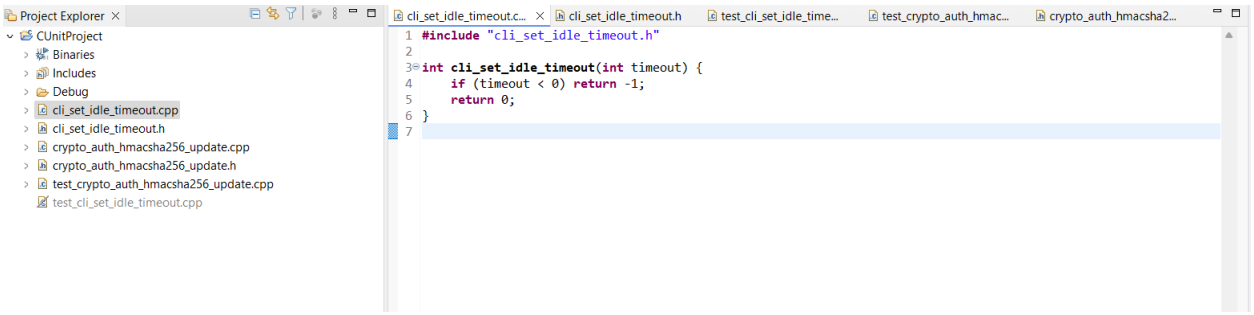
Set 7: Function Testing

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)



- crypto_auth_hmacsha256_update(const char*, size_t)

☑ 2. Test Scripts

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

cli_set_idle_timeout.c... | cli_set_idle_timeout.h | test_cli_set_idle_time... | test_crypto_auth_hmac... | crypto_auth_hmacsha2...

```c
#include <CUnit/CUnit.h>
#include <CUnit/Basic.h>
#include "cli_set_idle_timeout.h"  // ☑ not the .cpp file

void test_valid_timeout() {
    CU_ASSERT(cli_set_idle_timeout(10) == 0);
}

void test_invalid_timeout() {
    CU_ASSERT(cli_set_idle_timeout(-5) == -1);
}

int main() {
    CU_initialize_registry();
    CU_pSuite suite = CU_add_suite("IdleTimeoutSuite", 0, 0);
    CU_add_test(suite, "Test valid timeout", test_valid_timeout);
    CU_add_test(suite, "Test invalid timeout", test_invalid_timeout);
    CU_basic_run_tests();
    CU_cleanup_registry();
    return 0;
}
```

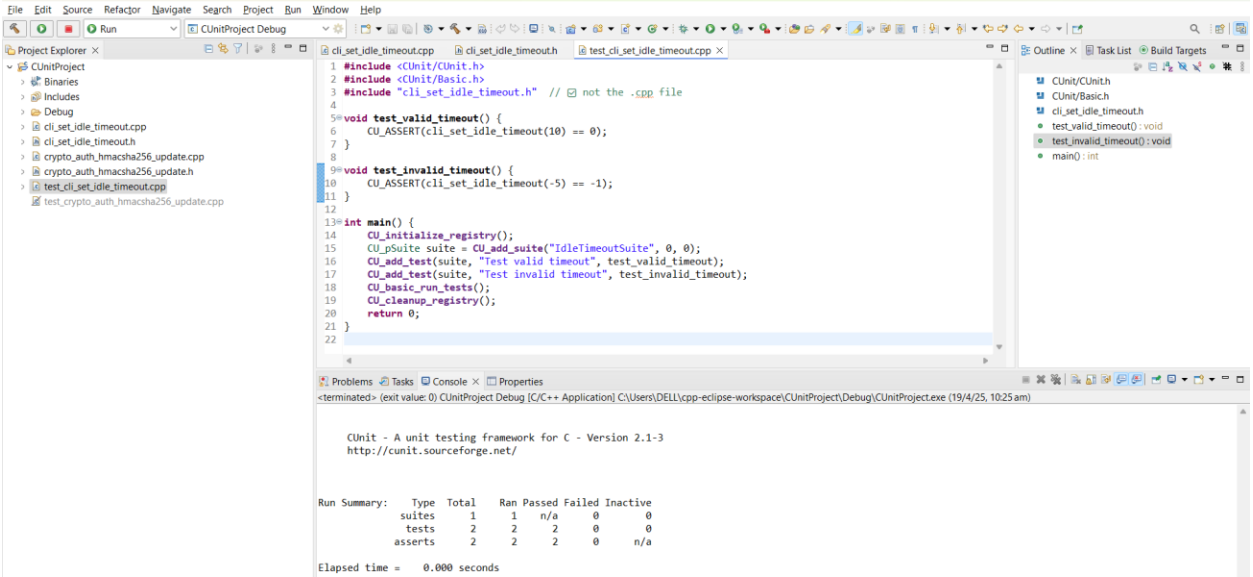
- Manual main-based test calls were created and verified for crypto_auth_hmacsha256_update().

cli_set_idle_timeout.c... | cli_set_idle_timeout.h | test_cli_set_idle_time... | test_crypto_auth_hmac... | crypto_auth_hmacsha2...

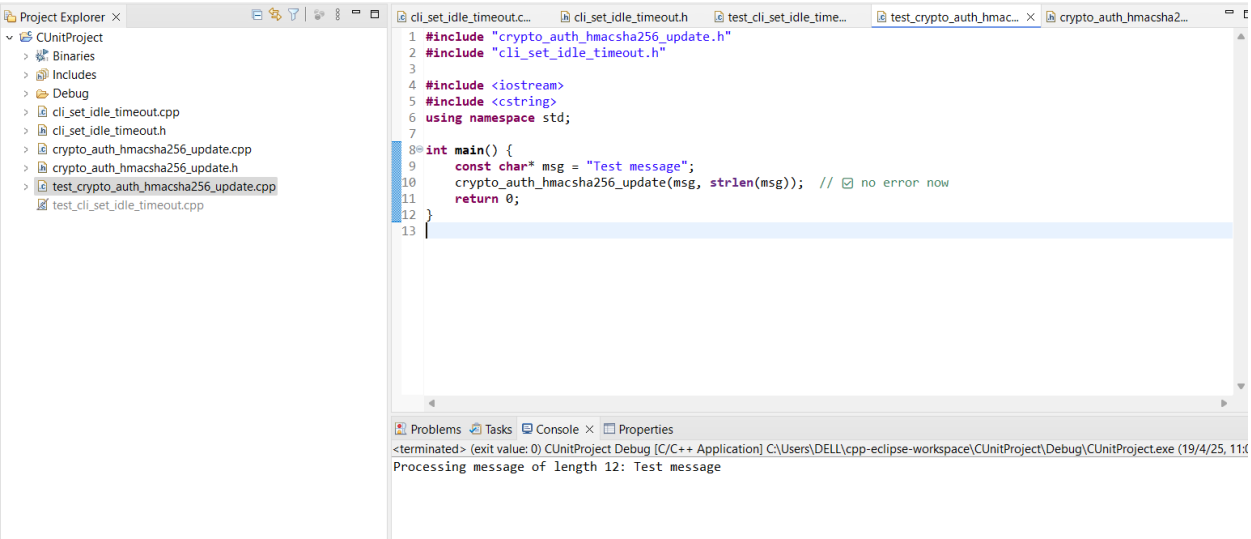
```c
#include "crypto_auth_hmacsha256_update.h"
#include "cli_set_idle_timeout.h"

#include <iostream>
#include <cstring>
using namespace std;

int main() {
    const char* msg = "Test message";
    crypto_auth_hmacsha256_update(msg, strlen(msg));  // ☑ no error now
    return 0;
}
```

3. Execution & Screenshot

• CUnit test cases for cli_set_idle_timeout executed successfully



• Console test output for crypto_auth_hmacsha256_update



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")

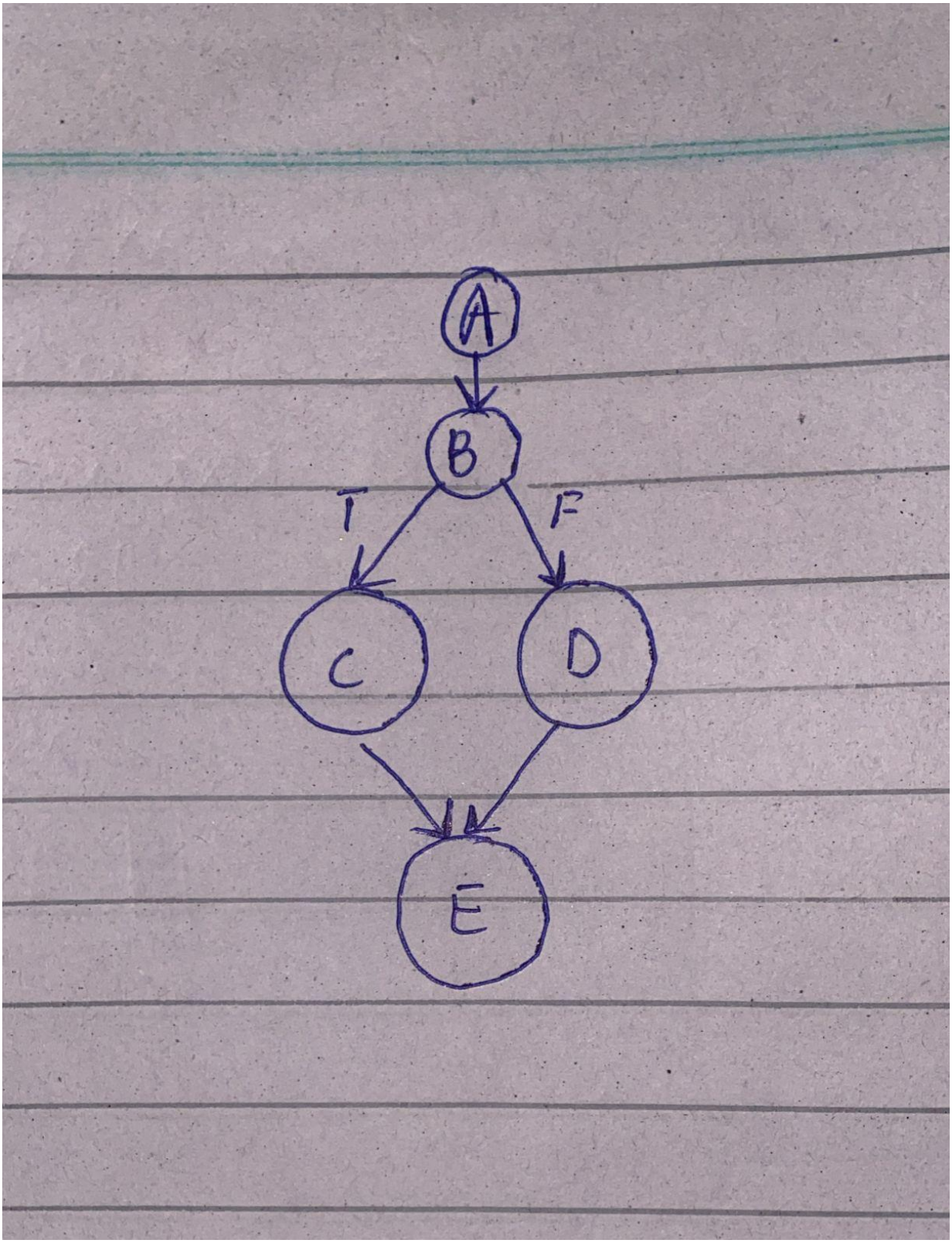
☑ 3. Edge-Pair Coverage (EPC) + Control Flow Graph (CFG)

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):


A
B
T
F
C
D
E

Edge-Pairs:

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

EPC Test Values:

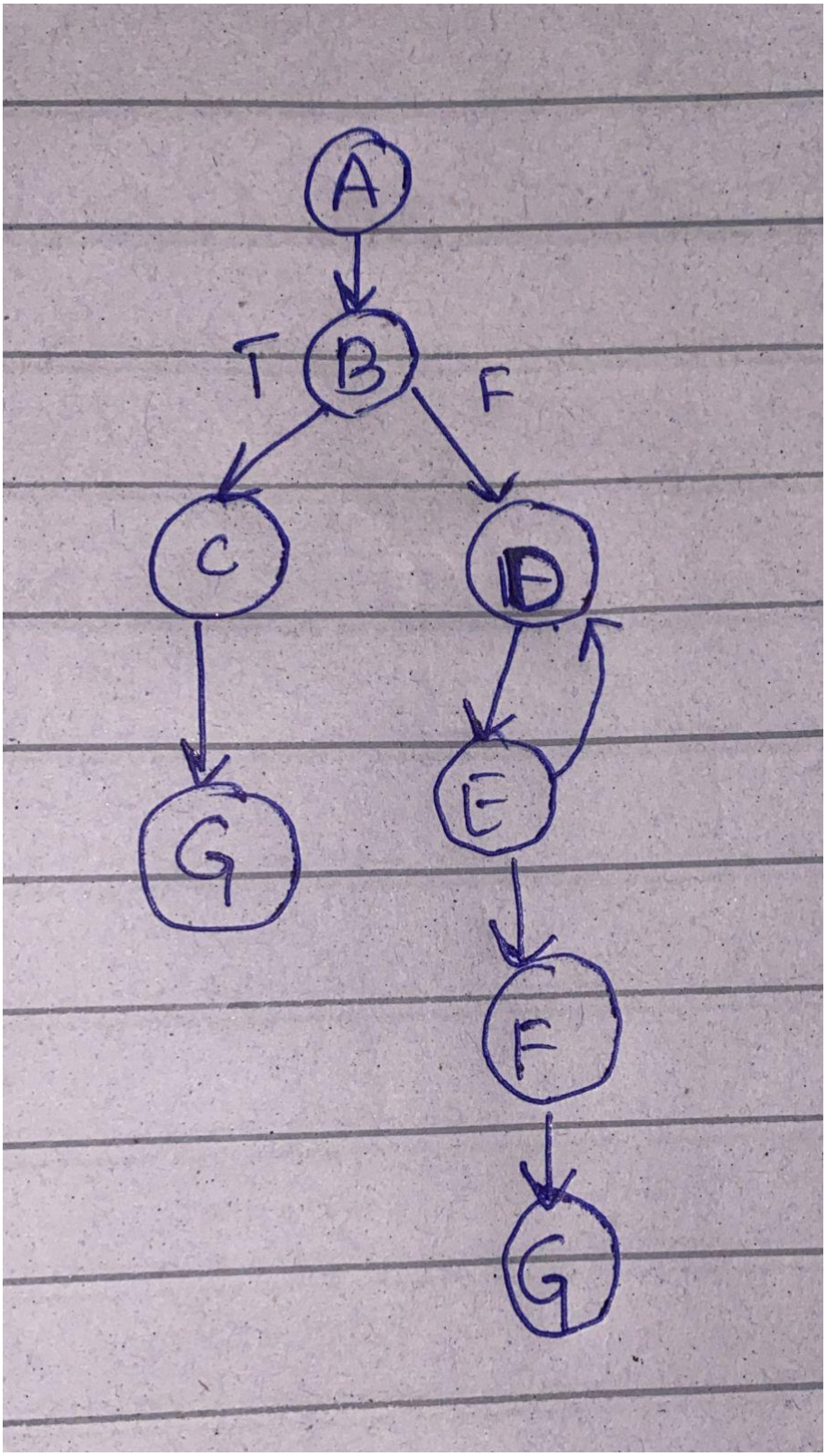
Edge-Pair	Test Value	Explanation
A → B → C → E	`-5`	Negative → returns `-1`
A → B → D → E	`10`, `0`	Valid → returns `0`

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 < length)
D → F	Loop exit when i ≥ 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 == length`	Ends printing, return `0`