Explain the steps to write a unit test case in Embedded C

1. Select a Unit Testing Framework
2. Make Code Testable
3. Set Up the Test Environment
    a. Build System Configuration
    b. Include Paths
    c. Mocking/Stubbing Setup
4. Write Test Cases
    a. Normal operation
    b. Edge cases
    c. Error conditions
    d. Assertions
5. Execute Tests
6. Analyze and Debug
7. Refactor and Repeat

Write a program for Validating Boolean

```
int a=10;
1. //This evaluates to pass
2. TEST_ASSERT( (a > 0) )
3. TEST_ASSERT_TRUE( (a > 0) )
4. TEST_ASSERT_UNLESS( (a == 0) )
5. TEST_ASSERT_FALSE( (a == 0) )
   //This evaluates to fail
6. TEST_ASSERT( (a == 0) )
7. TEST_ASSERT_TRUE( (a == 0) )
8. TEST_ASSERT_UNLESS( (a >> 0) )
9. TEST_ASSERT_FALSE( (a >> 0) )
```

Write a program for Validating Integers

```
int a=10;
//This will evaluates to pass
1. TEST_ASSERT_EQUAL_INT(10, a);
   //This will evaluates to fail
2. TEST_ASSERT_EQUAL_INT(5, a);
```

Write a program for Validating bits

1. int act = 0x12FF;
2. TEST_ASSERT_BITS(0xF, 0xFFFFFFFF, act);
3. TEST_ASSERT_BITS(0xF00, 0xFFFFFFFF, act);
4. TEST_ASSERT_BITS_HIGH(0xF, act);
5. TEST_ASSERT_BITS_LOW(0xF00, act);
6. TEST_ASSERT_BIT_HIGH(0, act);
7. TEST_ASSERT_BIT_LOW(8, act);

Write a program for Validating Structures and memory

```
typedef struct{
   int a;
   int b;
} temp;
temp a_struct = {10, 20};
temp b_struct = {10, 20};
temp c_struct = {5, 20};
char a[] = "Embetroncix";
char *b = a;
char c[] = "embetronicx";
//This will pass
```

1. TEST_ASSERT_EQUAL_MEMORY(a, b, 5);
   //This will pass
2. TEST_ASSERT_EQUAL_MEMORY(&a_struct, &b_struct, sizeof(temp));
   //This will fail
3. TEST_ASSERT_EQUAL_MEMORY(&a_struct, &c_struct, sizeof(temp));
   //This will fail
4. TEST_ASSERT_EQUAL_MEMORY(a, c, 5);

Write a program for Validating Arrays

```
unsigned int a0[] = {1, 8, 567, 120};
unsigned int a1[] = {1, 8, 567, 120};
unsigned int a2[] = {1, 8, 567, 10};

//These below cases will pass
```

1. TEST_ASSERT_EQUAL_INT_ARRAY(a0, a1, 4);
2. TEST_ASSERT_EQUAL_INT_ARRAY(a0, a2, 3);
3. TEST_ASSERT_EQUAL_UINT_ARRAY(a0, a1, 4);
4. TEST_ASSERT_EQUAL_HEX32_ARRAY(a0, a1, 4);

   //These below cases will fail
5. TEST_ASSERT_EQUAL_INT_ARRAY(a0, a2, 4);
6. TEST_ASSERT_EQUAL_INT_ARRAY(a0, a2, 4);
7. TEST_ASSERT_EQUAL_UINT_ARRAY(a0, a2, 4);
8. TEST_ASSERT_EQUAL_HEX32_ARRAY(a0, a2, 4);

Write a program for _MESSAGE variant

int a=10;

//This will evaluates to fail and print the message

1. TEST_ASSERT_EQUAL_INT_MESSAGE(13, a, "Test Failed: \"a\" should be 13");

OUTPUT :: "Expected 13 Was 10. Test Failed: "a" should be 13"