

## CSE 5382 Secure Programming Fall 2018, Programming Assignment 2

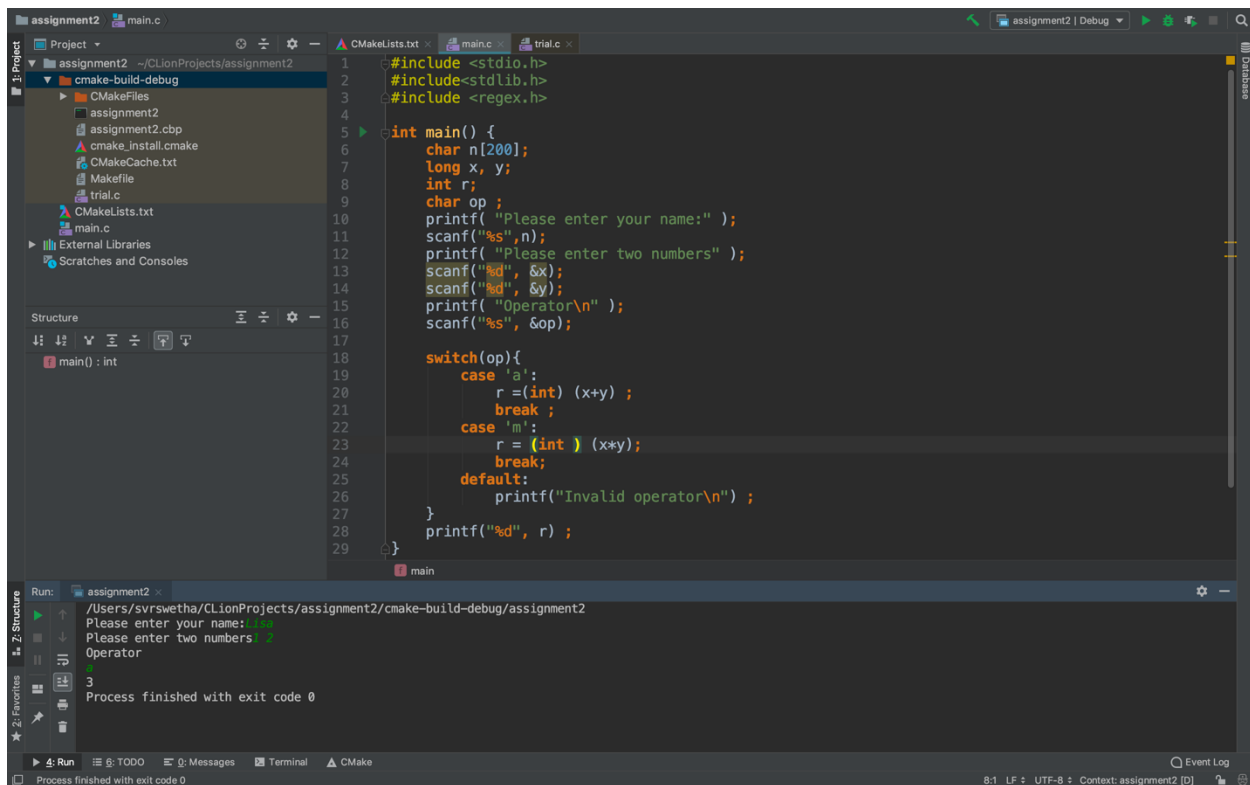
### Simple Calculator Swetha, Vijaya Raghavan | sxv1229| 1001551229

## Tool Versions

The tools I have used to test the code are: CLion 2018.2.4, Flaw finder

## Step 1:

Initial C Code with slight modification in order to make the code run.



```
1 #include <stdio.h>
2 #include<stdlib.h>
3 #include <regex.h>
4
5 int main() {
6     char n[200];
7     long x, y;
8     int r;
9     char op ;
10    printf( "Please enter your name:" );
11    scanf("%s",n);
12    printf( "Please enter two numbers" );
13    scanf("%d", &x);
14    scanf("%d", &y);
15    printf( "Operator\n" );
16    scanf("%s", &op);
17
18    switch(op){
19        case 'a':
20            r =(int) (x+y) ;
21            break ;
22        case 'm':
23            r = (int) (x*y);
24            break;
25        default:
26            printf("Invalid operator\n" );
27    }
28    printf("%d", r) ;
29 }
```

Run: assignment2 x

/Users/svswetha/CLionProjects/assignment2/cmake-build-debug/assignment2

Please enter your name:

Please enter two numbers:

Operator

3

Process finished with exit code 0

## Step 2: Results of Flaw finder tool on Step 1.

```
Swethas-MacBook-Pro:~ svrswetha$ flawfinder CLionProjects/assignment2/main.c
Flawfinder version 2.0.6, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining CLionProjects/assignment2/main.c

FINAL RESULTS:

CLionProjects/assignment2/main.c:11: [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification, permits
  buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or use a
  different input function.
CLionProjects/assignment2/main.c:16: [4] (buffer) scanf:
  The scanf() family's %s operation, without a limit specification, permits
  buffer overflows (CWE-120, CWE-20). Specify a limit to %s, or use a
  different input function.
CLionProjects/assignment2/main.c:6: [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.

ANALYSIS SUMMARY:

Hits = 3
Lines analyzed = 28 in approximately 0.01 seconds (5527 lines/second)
Physical Source Lines of Code (SLOC) = 27
Hits@level = [0] 7 [1] 0 [2] 1 [3] 0 [4] 2 [5] 0
Hits@level+ = [0+] 10 [1+] 3 [2+] 3 [3+] 2 [4+] 2 [5+] 0
Hits/KSLOC@level+ = [0+] 370.37 [1+] 111.111 [2+] 111.111 [3+] 74.0741 [4+] 74.0741 [5+] 0
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://www.dwheeler.com/secure-programs) for more information.
```

## Step 3: Fixed the above code In C language: (Using "better" string handling, integer management, and handle "international" characters.)

```
//
// Created by Swetha Vijaya Raghavan on 10/7/18.
//

#include<stdio.h>
#include <stdlib.h>
#include <regex.h>
#define BUFFER_SIZE 256

char* get_user(char str[]){
    regex_t regex ;
    regcomp(&regex,"^[a-z A-z]",0);
    while( 1 ){
        printf("Enter a name : ");
        fgets(str, sizeof(char)*50, stdin) ;
        if(!regexec(&regex, str, 0, NULL, 0))
            break ;
        printf("Invalid Name\n") ;
    }
    return str ;
}

long get_number(char t, char str[]){
```

```

    regex_t regex ;
    regcomp(&regex,"^[0-9]",0);
    long x = 0;
    char *end ;
    while( 1 ){
        printf("Enter a number %c :", t) ;
        fgets(str, sizeof(char)*50, stdin) ;
        if(!regexec(&regex, str, 0, NULL, 0)){
            break ;
        }else{
            printf("Invalid number\n") ;
        }
    }
    x = strtol(str, &end, 10) ;
    return x ;
}

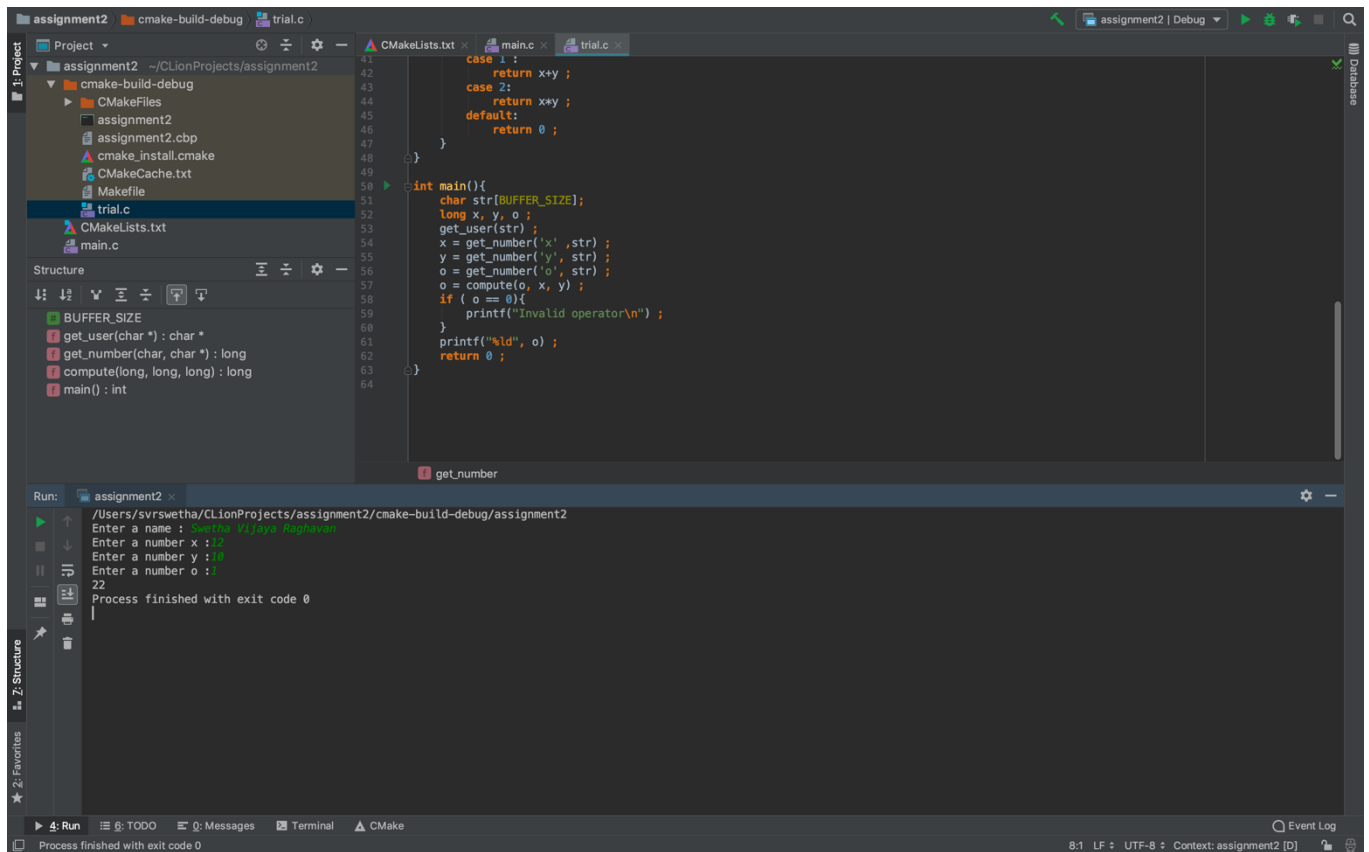
long compute(long o, long x, long y){
    switch(o){
        case 1 :
            return x+y ;
        case 2:
            return x*y ;
        default:
            return 0 ;
    }
}

int main(){
    char str[BUFFER_SIZE];
    long x, y, o ;
    get_user(str) ;
    x = get_number('x', str) ;
    y = get_number('y', str) ;
    o = get_number('o', str) ;
    o = compute(o, x, y) ;
    if ( o == 0){
        printf("Invalid operator\n") ;
    }
    printf("%ld", o) ;
    return 0 ;
}

```

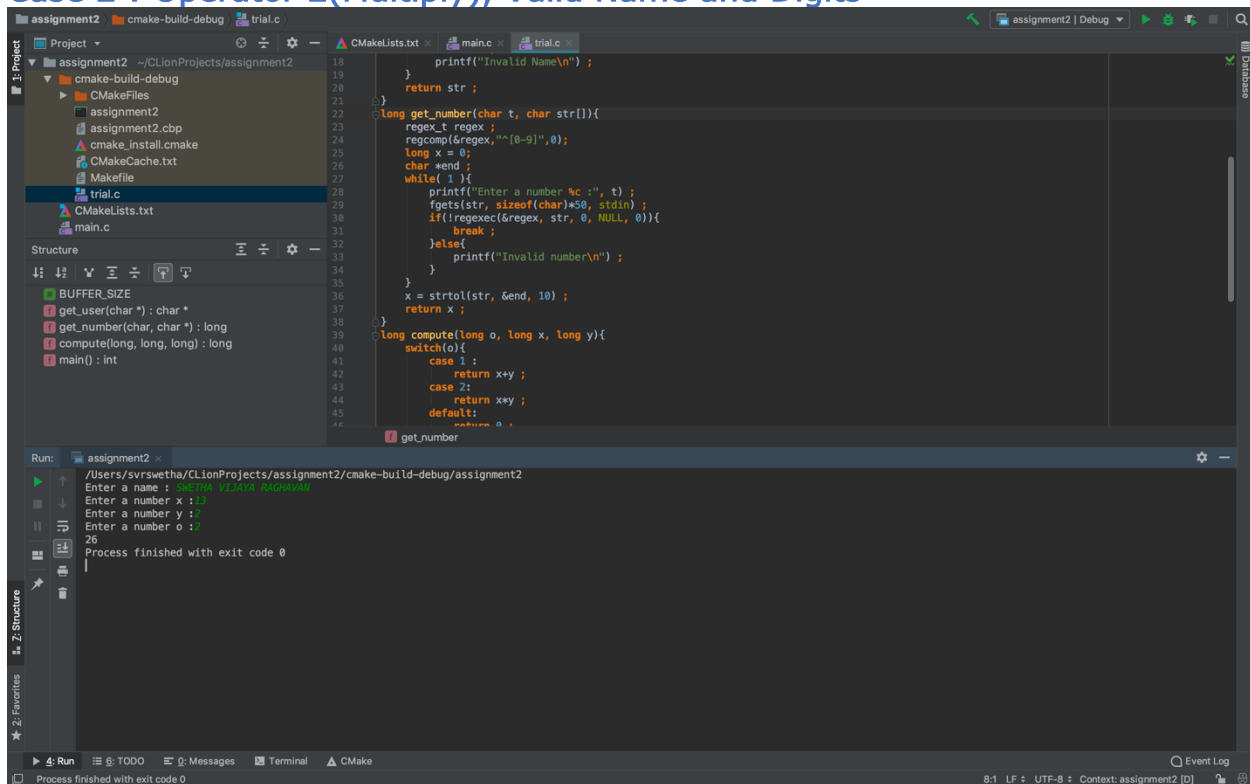
Step 4: Screenshots of the Output of different Cases:(operator – 1 add, operator – 2 multiply rest Invalid operator)

## Case 1: Operator 1(add), Valid Name and Digits



## Case 1 Screenshot

## Case 2 : Operator 2(Multiply), Valid Name and Digits



The screenshot displays a CMake IDE interface. The left sidebar shows the project structure for 'assignment2', including files like 'assignment2.cbp', 'cmake\_install.cmake', 'CMakeCache.txt', 'Makefile', 'trial.c', 'CMakeLists.txt', and 'main.c'. The main editor window shows the source code for 'trial.c'. The code includes a regex for validating names (only lowercase letters and spaces) and a loop for getting a number. The 'main' function calls 'get\_user' to get a name, 'get\_number' to get a number, and 'compute' to perform the operation. The 'compute' function uses a switch statement to handle addition and multiplication. The 'get\_number' function uses a while loop to ensure a valid number is entered. The bottom panel shows the execution output, which matches the test case description: 'Enter a name : invalid invalid', 'Enter a number x : 7', 'Enter a number y : 1', 'Enter a number o : 26', and 'Process finished with exit code 0'.

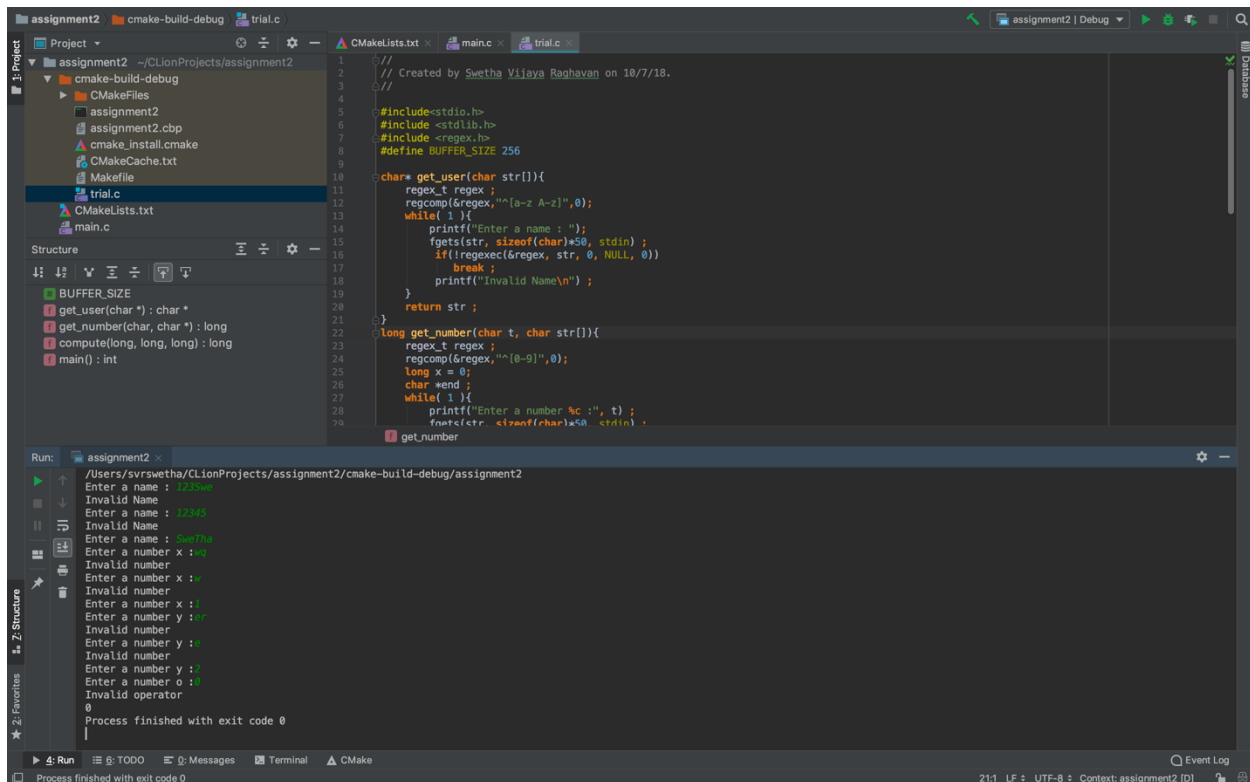
```
18 printf("Invalid Name\n");
19 }
20 return str ;
21 }
22 long get_number(char t, char str[]){
23     regex_t regex ;
24     regcomp(&regex, "^[a-z ]*", 0);
25     long x = 0;
26     char *end ;
27     while(1){
28         printf("Enter a number %c :", t) ;
29         fgets(str, sizeof(char)*50, stdin) ;
30         if(!regexec(&regex, str, 0, NULL, 0)){
31             break ;
32         }else{
33             printf("Invalid number\n") ;
34         }
35     }
36     x = strtol(str, &end, 10) ;
37     return x ;
38 }
39 long compute(long o, long x, long y){
40     switch(o){
41         case 1 :
42             return x+y ;
43         case 2:
44             return x*y ;
45         default:
46             return 0 ;
47     }
48 }
```

Run: assignment2 x  
/Users/svrswetha/CLionProjects/assignment2/cmake-build-debug/assignment2  
Enter a name : invalid invalid  
Enter a number x : 7  
Enter a number y : 1  
Enter a number o : 26  
Process finished with exit code 0

## Case 2 Screenshot

## Case 3:

- Tried Giving digits In Names, combination of digits & letters, throws Invalid Name then entered a valid name (regex takes small letter, Capital Letters space in between)
- Gave letters instead of digits in Numbers. Throws Invalid Number.
- Gave 0 for Operator throws Invalid Operator i.e., Other than 1 or 2;



## Case 3 Screenshot

Step 5: Results of Flaw finder tool on the modified C code. Handled: "better" string handling, integer management, and handle "international" characters.

```

[Swethas-MacBook-Pro:~ svrswetha$ flawfinder CLionProjects/assignment2/main.c
Flawfinder version 2.0.6, (C) 2001-2017 David A. Wheeler.
Number of rules (primarily dangerous function names) in C/C++ ruleset: 223
Examining CLionProjects/assignment2/main.c

FINAL RESULTS:

CLionProjects/assignment2/main.c:51: [2] (buffer) char:
  Statically-sized arrays can be improperly restricted, leading to potential
  overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
  functions that limit length, or ensure that the size is larger than the
  maximum possible length.

ANALYSIS SUMMARY:

Hits = 1
Lines analyzed = 63 in approximately 0.01 seconds (11864 lines/second)
Physical Source Lines of Code (SLOC) = 57
Hits@level = [0] 6 [1] 0 [2] 1 [3] 0 [4] 0 [5] 0
Hits@level+ = [0+] 7 [1+] 1 [2+] 1 [3+] 0 [4+] 0 [5+] 0
Hits/KSLOC@level+ = [0+] 122.807 [1+] 17.5439 [2+] 17.5439 [3+] 0 [4+] 0 [5+] 0
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming HOWTO'
(https://www.dwheeler.com/secure-programs) for more information.

```

- Hits reduced to 1. Scanf buffer flow is handled.

## Step 6: Rust Code

```
main.rs
1 extern crate regex;
2 fn main(){
3     use std::io::{stdin, stdout, Write};
4     use regex::Regex ;
5     let re = Regex::new(r"^[a-zA-Z\s\.\,]*$").unwrap() ;
6     let mut s:String::new();
7     let _=stdout().flush();
8     while true{
9         s.clear() ;
10        print!("Please Enter a name :") ;
11        let _=stdout().flush() ;
12        stdin().read_line(&mut s).expect("Did not enter a correct string");
13        print!("{}", s);
14        if re.is_match(&s){
15            break ;
16        }
17        println!("Name is invalid");
18        let _=stdout().flush();
19    }
20    println!("Enter a operator") ;
21    let _= stdout().flush() ;
22    let mut num = String::new();
23    stdin().read_line(&mut num).expect("Enter a number") ;
24    println!("Enter two numbers") ;
25    let _= stdout().flush() ;
26    let (mut x, mut y) = (String::new(), String::new());
27    print!("Enter a number : ") ;
28    let _= stdout().flush() ;
29    stdin().read_line(&mut x).expect("Enter a number") ;
30    print!("Enter another number :") ;
31    let _= stdout().flush() ;
32    stdin().read_line(&mut y).expect("Enter a number") ;
33    match num.trim().parse::<u32>().unwrap()*y.trim().parse::<u32>().unwrap(),
34        1 => println!("{}", x.trim().parse::<u32>().unwrap()*y.trim().parse::<u32>().unwrap()),
35        2 => println!("{}", x.trim().parse::<u32>().unwrap()+y.trim().parse::<u32>().unwrap()) ,
36        _ => println!("Invalid operator") ,
37    }
38 }
39
```

## Step 7: Results of Step 5 (Rust Code)

```
Swethas-MacBook-Pro:debug svrswetha$ cargo build
Finished dev [unoptimized + debuginfo] target(s) in 0.04s
Swethas-MacBook-Pro:debug svrswetha$ ./hel
Please Enter a name :swe123
swe123
Name is invalid
Please Enter a name :123we
123we
Name is invalid
Please Enter a name :s we
s we
Enter a operator
2
Enter two numbers
Enter a number : 1
Enter another number :2
2
Swethas-MacBook-Pro:debug svrswetha$ ./hel
Please Enter a name :swe .tha
swe .tha
Enter a operator
2
Enter two numbers
Enter a number : 1
Enter another number :2
2
```