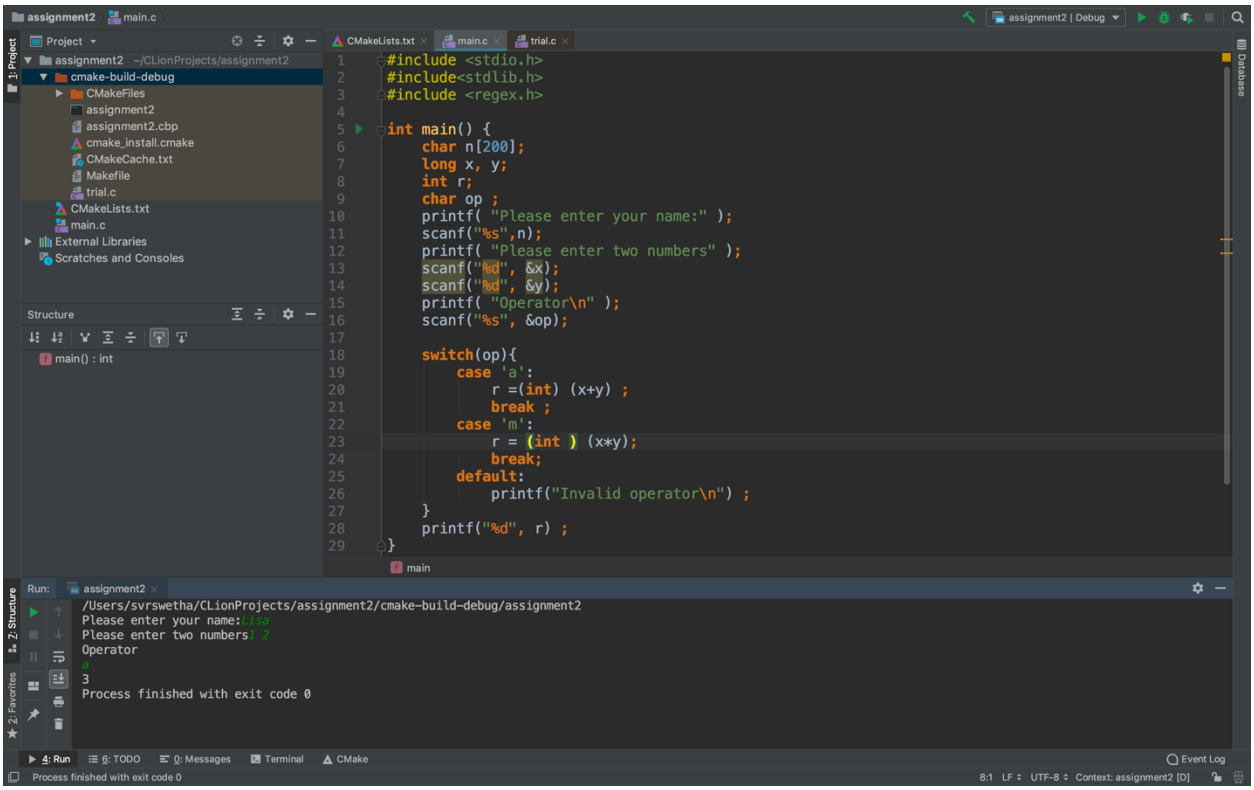
CSE 5382 Secure Programming Fall 2018, Programming Assignment 2
Simple Calculator Swetha, Vijaya Raghavan | sxv1229| 1001551229

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.



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.
```

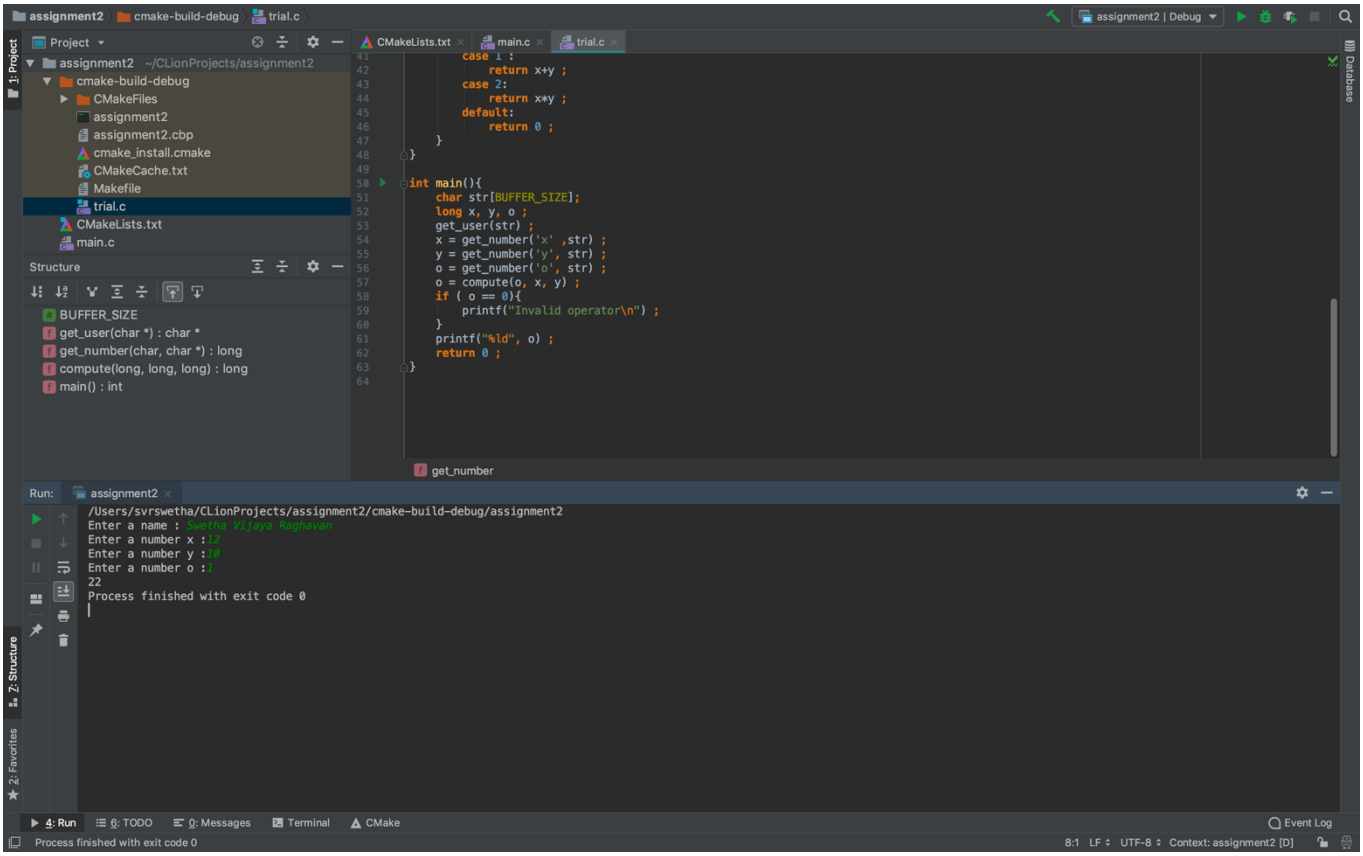
Step 3: Fixed the above code In C language: (Using "better" string handling, integer management, and handle "international" characters.)

```c
//
// 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[]){
```

```c
    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 ;
}
```

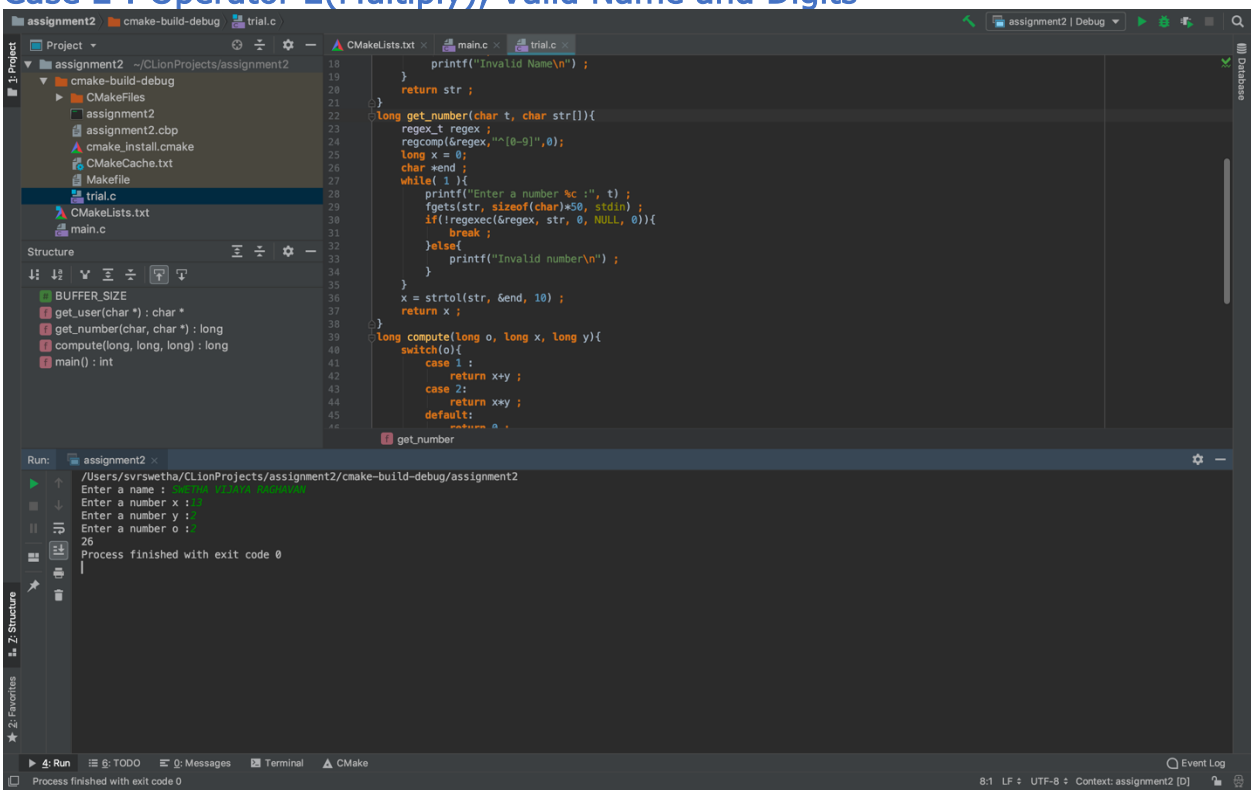
Step 4: Screenshots of the Output of different Cases:(operator – 1 add, operator – 2 multiply rest Invalid operator)

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



Case 1 Screenshot

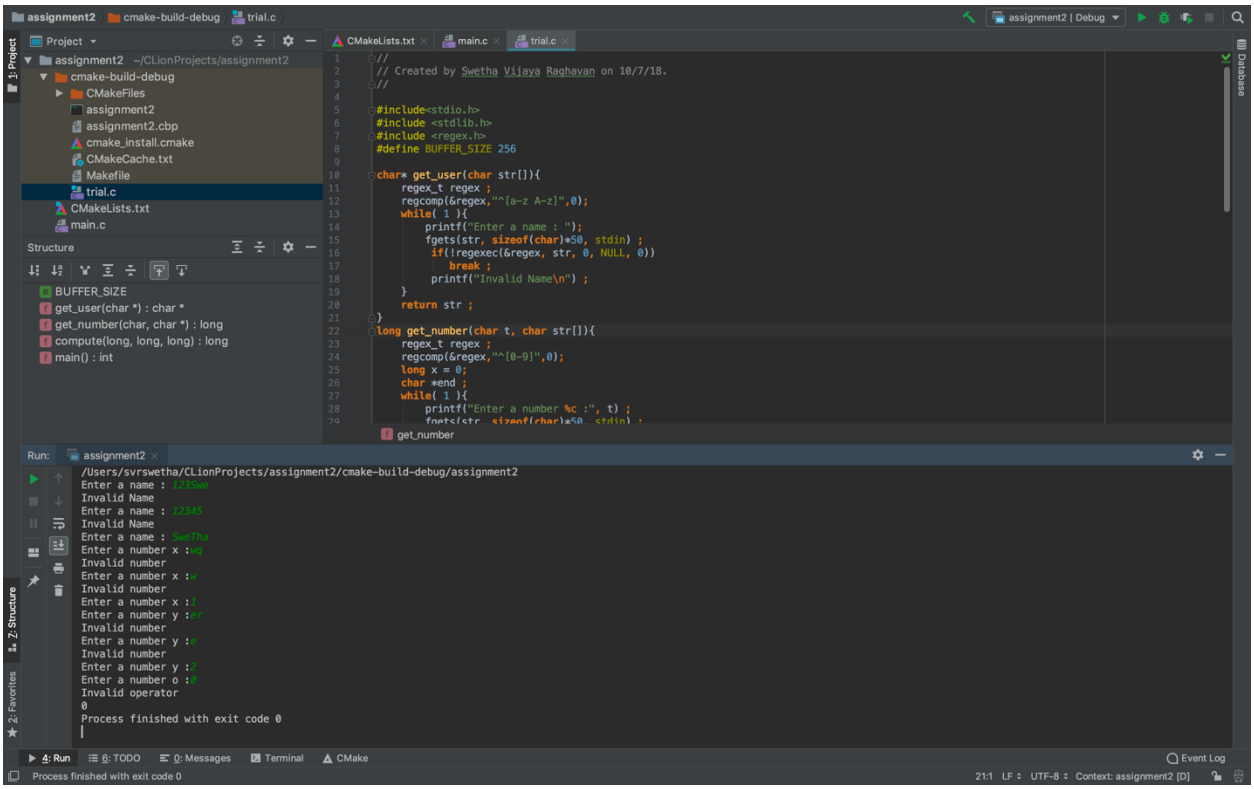
Case 2 : Operator 2(Multiply), Valid Name and Digits



Case 2 Screenshot

Case 3:

i. 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)
ii. Gave letters instead of digits in Numbers. Throws Invalid Number.
iii. 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

```rust
extern crate regex;
fn main(){
    use std::io::{stdin,stdout,Write};
    use regex::Regex ;
    let re = Regex::new(r"^[a-zA-Z\s\.]+$").unwrap() ;
    let mut s=String::new();
    let _=stdout().flush();
    while true{
        s.clear() ;
        print!("Please Enter a name :") ;
        let _=stdout().flush() ;
        stdin().read_line(&mut s).expect("Did not enter a correct string");
        print!("{}", s);
        if re.is_match(&s){
            break ;
        }
        println!("Name is invalid");
        let _ =stdout().flush();
    }
    println!("Enter a operator") ;
    let _ = stdout().flush() ;
    let mut num = String::new();
    stdin().read_line(&mut num).expect("Enter a number") ;
    println!("Enter two numbers") ;
    let _ = stdout().flush() ;
    let (mut x, mut y) = (String::new(), String::new());
    print!("Enter a number : ") ;
    let _ = stdout().flush() ;
    stdin().read_line(&mut x).expect("Enter a number") ;
    print!("Enter another number :") ;
    let _ = stdout().flush() ;
    stdin().read_line(&mut y).expect("Enter a number") ;
    match num.trim().parse::<u32>().unwrap(){
        1 => println!("{}", x.trim().parse::<u32>().unwrap()+y.trim().parse::<u32>().unwrap()),
        2 => println!("{}", x.trim().parse::<u32>().unwrap()*y.trim().parse::<u32>().unwrap()) ,
        _ => println!("Invalid operator") ,
    }
}
```

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
```