

Lab 1 Report

Zhaoyi Wang 1689747

Deliverable 1

```
hello_world on  master [?] is 📦 v0.1.0 via 🦀 v1.54.0 via © base
→ cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.02s
    Running `target/debug/hello_world`
Hello, world!
```

Deliverable 2

```
use rand::Rng;

fn main() {
    let secret_number_x: i32 = rand::thread_rng().gen();
    let secret_number_y = rand::thread_rng().gen_range(-10.0..10.0);

    println!("x is {}", secret_number_x);
    println!("y is {}", secret_number_y);

    println!("A number between 0 and 9 is {}",
rand::thread_rng().gen_range(0..10));
}
```

The result is:

```
random_num on  master [?] is  v0.1.0 via  v1.54.0 via  base
→ cargo run
    Compiling random_num v0.1.0
(/Users/wangzhaoyi/RustProject/RustLab/Lab1/random_num)
    Finished dev [unoptimized + debuginfo] target(s) in 0.87s
    Running `target/debug/random_num`
x is -1921194873
y is 9.408560585929344
A number between 0 and 9 is 6
```

Deliverable 3

Question 3-1

```
prime_num on  master [?] is  v0.1.0 via  v1.54.0 via  base
→ cargo run
    Compiling prime_num v0.1.0
(/Users/wangzhaoyi/RustProject/RustLab/Lab1/prime_num)
    Finished dev [unoptimized + debuginfo] target(s) in 0.50s
    Running `target/debug/prime_num`
candidate:74
After set:75
```

From the result, we can see that the target is to add 1 if a candidate number is a **even number**. To be specific, this function will give a random integer number between $[0, n]$, n is a number that set by the user. Then, it will set the least significant bit ($index=0$) to 1 ($true=1$ and $false=0$). This means that if it is an even number, it will become an odd number. Finally, the `is_prime()` function will decide whether it is a prime number. If the answer is yes, `is_prime()` will return this number.

Question 3-2

```
use prime_tools::*;
use rug::{Assign, Integer};
use rand::Rng;
```

```

fn main() {
    println!("Return value: {}", function(12345));
}

fn function(n: u32) -> Integer{
    let mut rng = rand::thread_rng();
    loop {
        let mut candidate = Integer::new();
        candidate.assign(rng.gen_range(0.. n));

        if is_u32_prime(candidate.to_u32().unwrap()) == true {
            return candidate;
        }
    }
}

```

The result is:

```

coder@ubuntu-s-1vcpu-2gb-tor1-
01:~/personalProj/rusttest/rustlab/prime_number$ cargo run
   Compiling prime_number v0.1.0
(/home/coder/personalProj/rusttest/rustlab/prime_number)
    Finished dev [unoptimized + debuginfo] target(s) in 1.07s
    Running `target/debug/prime_number`
Return value: 3301

```


Question 3-3

We define a function that has a `Integer` type variable parameter and a `bool` type return value. In this function, we define two variables called `s`, `d`, and the value of them comes from `rewrite()` function. In a loop which will run five times, a variable called `basic` will get value from the random number generator. Also, `x` equals to the function `mod_exp()`. Then, if `x` satisfy the condition, the iteration will continue. If `x` satisfy the condition in all five


iteration, the function will return `true` . Otherwise, a new for loop will be implemented. In this for loop, a value from `mod_exp()` will be assigned to the `x`. If `x` equals to `1_usize` , return `false` . If `x` equals to `candidate - 1_usize` , the for loop will break. If `i` for this loop equals to `s - 2_usize` , return `false` .


Question 3–4

The answer is **glass-pumpkin**.




 **crates.io**

[Browse All Crates](#) | [Docs ▼](#) | [Log in with GitHub](#)

 **Search Results for 'Prime and Prejudice'**

Displaying **1-1** of **1** total results Sort by  Relevance ▼

glass_pumpkin v1.2.0
A cryptographically secure prime number generator based on rust's own num-bigint and num-integer
[Homepage](#) [Documentation](#) [Repository](#)

 [All-Time: 7,991](#)
 [Recent: 1,294](#)
 [Updated: about 15 hours ago](#)

Question 3–5

```
use prime_tools::*;
use std::vec::Vec;

fn main() {
    let prime_list = find_consecutive_primes(1000);
    println!("The consecutive prime below 100 contains {} terms and
equal to {}", prime_list.1.len(), prime_list.0);
    println!("The list is: {:?}", prime_list.1);
}

// do the sum operation for the selected vector.
fn do_sum(vector: &Vec<u32>) -> u32 {
```

```

    let mut sum = 0;
    for num in &*vector {
        sum = num + sum;
    }
    sum
}

// use this function to find the consecutive primes we need.
// there are two conditions: Is the sum of the list < max? Is the sum of
the list prime?
fn find_consecutive_primes(max: u32) -> (u32, Vec<u32>) {
    let mut result_list = vec![2];

    // add all possible prime numbers to the list.
    for number in 3..max + 1 {
        if do_sum(&result_list) < max {
            if is_u32_prime(number)==true {
                result_list.push(number);
            }
        };
    }

    // find the appropriate slice of vector
    // e.g. [1,2,3,4,5]->[{1,2,3,4},5] [1,{2,3,4,5}] [{1,2,3},4,5] [1,
{2,3,4},5]...
    let mut final_prime_list: Vec<u32> = Vec::new();
    let mut right_index = result_list.len();
    while right_index != 0 {
        let mut left_index = 0;
        while (left_index + right_index) <= result_list.len() {
            // test all possible slice of vector
            let cur_list = result_list[left_index..(left_index +
right_index)].to_vec();
            // whether the sum is less than max?
            if do_sum(&cur_list) < max {
                // whether the sum is a prime number?

```

```

        if is_u32_prime(do_sum(&cur_list)) {
            // to find the longest list
            if cur_list.len() > final_prime_list.len() {
                final_prime_list = cur_list;
            }
        }
        left_index += 1;
    };
    right_index -= 1;
};
(do_sum(&final_prime_list), final_prime_list)
}

```

The result is:

```

/Users/wangzhaoyi/.cargo/bin/cargo run --color=always --package
find_prime --bin find_prime
    Finished dev [unoptimized + debuginfo] target(s) in 0.01s
    Running `target/debug/find_prime`
The consecutive prime below 100 contains 21 terms and equal to 953
The list is: [7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
67, 71, 73, 79, 83, 89]

```