

# CS6013: Advanced Data Structures and Algorithms

Programming Assignment III (out of 10 marks)

(Start Date: 11 November 2021)

(Submission Deadline: 11:59 pm, Sunday, 21 November 2021)

## 0.1 Miller-Rabin\_Test( $n$ )

**Input:** An odd integer  $n \geq 3$ .

**Output:** If  $n$  is prime, the algo always returns “prime”. If  $n$  is composite, the algo with probability at least  $1/2$  returns “composite”.

**Algo:**

STEP 0: Check if  $n = a^b$  for integers  $a, b \geq 2$ . If so, return “composite”.

STEP 1: Select  $a \in \{1, 2, \dots, n-1\}$  uniformly at random. Compute  $a^{n-1} \bmod n$ . If this is not 1, then return “composite”.

STEP 2: Let  $n-1 = 2^k t$ , where  $t$  is odd. Compute  $a^t \bmod n, a^{2t} \bmod n, a^{4t} \bmod n, a^{8t} \bmod n, \dots$ , until a 1 is seen. If the number before 1 is not  $-1$ , then return “composite”; else return “prime”.

Implement the function `Miller-Rabin_test( $n$ )` described above. Define a function `higher_power( $a, b$ )` that computes and returns  $a^b$  in  $\text{polylog}(n)$  time, where  $0 \leq a, b \leq n$  [Do not use any built-in function to compute  $a^b$  or  $a^b \bmod n$ ]. Invoke this function  $\text{polylog}(n)$  times to check whether  $n = a^b$  in STEP 0. You may use a standard library function to find a random number from the set  $\{1, 2, \dots, n\}$  in STEP 1. Define another function `modular_higher_power( $a, b, n$ )` that computes and returns  $a^b \bmod n$  in  $\text{polylog}(n)$  time,  $0 \leq a, b \leq n$ . Use this function to compute  $a^{n-1} \bmod n$  in STEP 1. Define a function `two_factorize( $x$ )` that computes and returns in  $\text{polylog}(n)$  time the non-negative integer  $y$  such that  $x = 2^y z$ , where  $z$  is odd and  $x \leq n$ . Invoke `two_factorize( $n-1$ )` to find  $k$  in STEP 2. Compute  $a^t \bmod n, a^{2t} \bmod n, a^{4t} \bmod n, \dots$  by repeatedly invoking the `modular_higher_power()` function with appropriate parameters.

In the main function, read positive integers  $n$  and  $r$ . Invoke the function `Miller-Rabin_Test( $n$ )`  $r$  times in a loop that runs from 1 to  $r$ . If all the  $r$  invocations of `Miller-Rabin_Test( $n$ )` return “prime”, then print “ $n$  is a prime number”. Else, print “ $n$  is a composite number”. We know that if  $n$  is actually prime, this algorithm prints “ $n$  is a prime number” with probability 1; if  $n$  is actually composite, this algorithm wrongly prints “ $n$  is a prime number” with probability at most  $\frac{1}{2^r}$ .

**Sample Output:**

$n = 12000$

$r = 25$

12000 is a composite number.

## 0.2 Program Related Instructions

1. You can write your program in one of C, C++, Java, or Python.

## 0.3 Submission Guidelines

1. Your submission will be one zip file named `<roll-number>.zip`, where you replace roll-number by your roll number (e.g. `cs20mtech11003.zip`), all in small letters. The compressed file should contain the below mentioned files:

- (a) Programming files (please do not submit python notebooks or IDE files). **The entire source code has to be in one file named `main_prog.c` (or `main_prog.cpp`, or ...).**
  - (b) **No need to submit a report.** However, if you wish you may submit a text/doc file giving a detailed description of your program. No marks for this.
  - (c) Upload your zip file in Google Classroom at Classwork→Week 13→Assignment 3. No delays permitted.
2. Failure to comply with instructions (file-naming, upload, input/output specifications) will result in your submission not being evaluated (and you being awarded 0 for the assignment).
3. **Plagiarism policy:** If we find a case of plagiarism in your assignment (i.e. copying of code, either from the internet, or from each other, in part or whole), you will be awarded a zero and will lead to a FR grade for the course in line with the department Plagiarism Policy (<https://cse.iith.ac.in/academics/plagiarism-policy.html>). Note that we will not distinguish between a person who has copied, or has allowed his/her code to be copied; both will be equally awarded a zero for the submission.

## 0.4 Evaluation Scheme

Your assignment will be awarded marks based on the following aspects:

- Code clarity (includes comments, indentation, naming of variables and functions, etc.): 1 mark
- Perfect output: 2 mark
- Logic in the code of functions `higher_power()`, `modular_higher_power()`, `two_factorize()`:  $1 + 1 + 1 = 3$  marks.
- Logic in the code of the function `Miller-Rabin_Test()`: 4 marks.