Test Name : Cohort1-Midterm-T2-AY2021-22-SM404-A-computational-introduction-to-Number-Theory

Name : Naitiksinh Dharmendrasinh Solanki - naitiksinh.solanki@iiitb.ac.in

Test Start Time	Marks Scored	Total Questions	Attempted Questions
10/3/2022, 4:00:53 PM	34.0 / 60.0	9	7

Correct Questions	Incorrect Questions	Skipped Questions	Pending Evaluation
7	0	2	0

List of Sections

Integers and Congruences

Marks per question : 5.0 | **Marks Scored : 10.0**

Q No.	Q. Type	Status	Marks	
1	File Upload	✔	1.0	Hide Answer

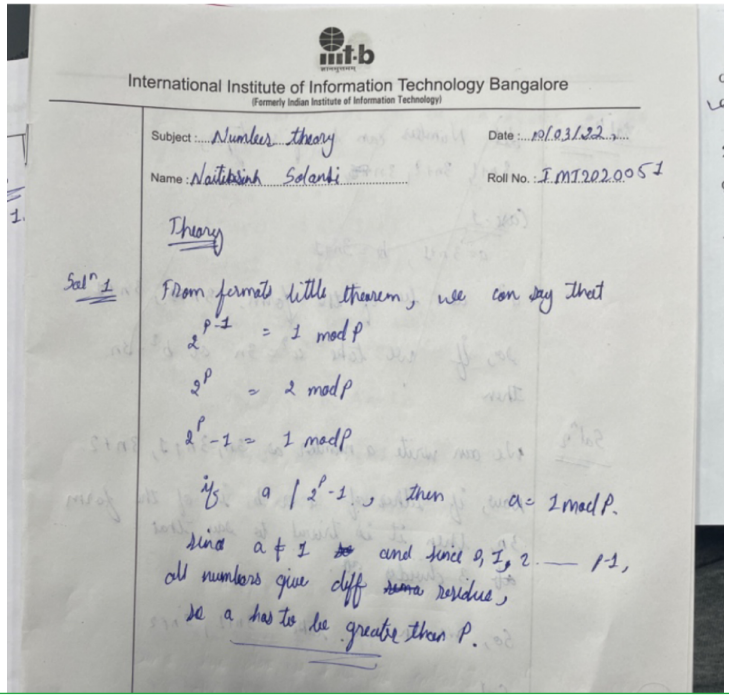
Let p be an odd prime number. Prove using Fermat's Little Theorem that every prime divisor of $2^p - 1$ is greater than p.

Uploaded File Details

Original Response

Previous Page: 1 / 1 Next

40 %



International Institute of Information Technology Bangalore
(Formerly Indian Institute of Information Technology)

Subject : Number theory Date : 29/03/22

Name : Naitikkumar Solanki Roll No. : IMT2020051

Theory

Sol^n 1

From fermat's little theorem, we can say that

$2^{p-1} = 1 \bmod p$

$2^p = 2 \bmod p$

$2^p - 1 = 1 \bmod p$

if $a \mid 2^p - 1$, then $a = 1 \bmod p$.

since $a \neq 1$ and since $0, 1, 2 \ldots p-1$, all numbers give diff remainder, so a has to be greater than p.

2	File Upload	✔	5.0	Hide Answer

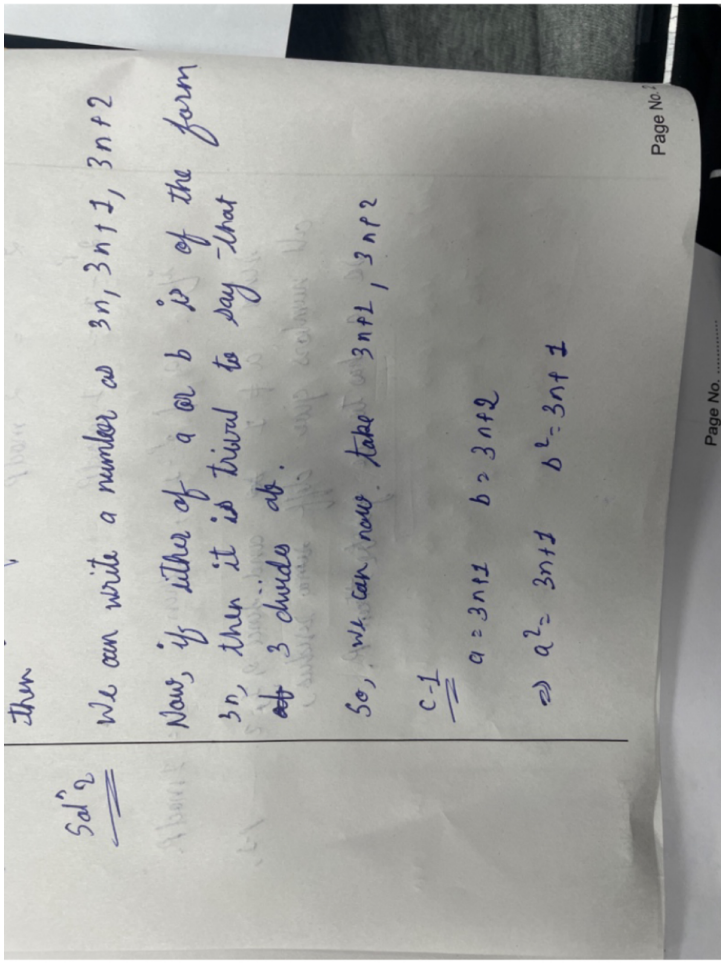
If $a^2 + b^2 = c^2$ where a, b, c are positive integers then show that 3 divides ab.

Uploaded File Details

Original Response

Previous Page: 1 / 2 Next

40 %



Previous Page: 1 / 2 Next

Evaluator Comments

Writing can be a bit more clear.

3	File Upload	✔	2.0	Hide Answer

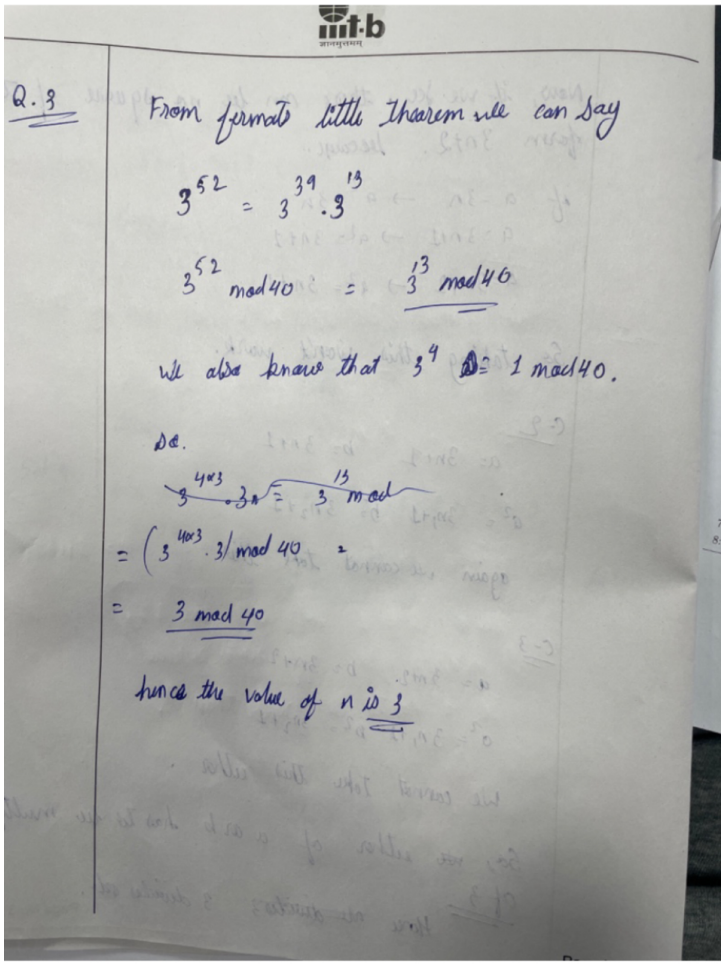
Find a value of n in $3^{52} \equiv n \mod 40$.

Uploaded File Details

Original Response

Previous Page: 1 / 1 Next

40 %



Previous Page: 1 / 1 Next

4 File Upload ⚠ 0.0 Hide Answer

It follows from the Chinese Remainder Theorem that there is an isomorphism

$$\phi : \frac{Z}{20Z} \to \frac{Z}{4Z} \times \frac{Z}{5Z}$$

In this case what is $\phi^{-1}(1, 3)$?

User did not upload files for this question

Evaluator Comments

Not Answered

5 File Upload ⚠ 0.0 Hide Answer

Let p and q be distinct odd primes then $p^{q-1} + q^{p-1} \equiv 1 \bmod pq$.

User did not upload files for this question

Evaluator Comments

Not Answered

6 File Upload ✔ 2.0 Hide Answer

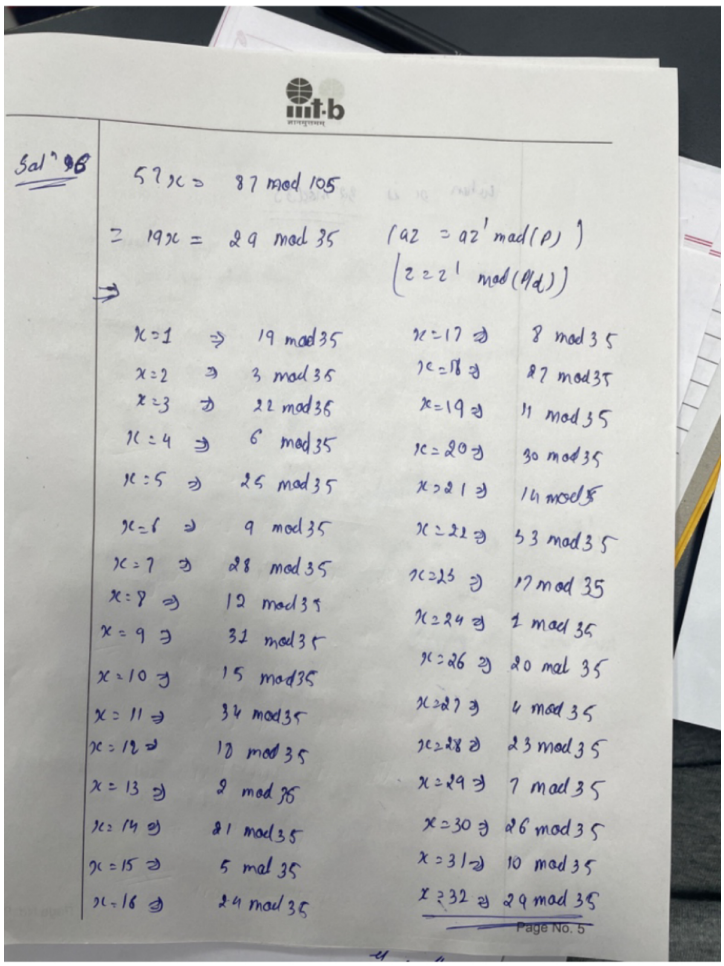
Find all solutions of the congruence $57\ X \equiv\ 87 \bmod 105$.

Uploaded File Details

Original Response

Previous Page: 1 / 2 Next

40 %



Previous Page: 1 / 2 Next

Algorithms Marks per question : 10.0 **Marks Scored : 24.0**

Q No.	Q. Type	Status	Marks	
1	File Upload	✔	7.0	Hide Answer

Show that the following algorithm (known as the *Repeated Squaring Algorithm*) correctly computes a^n for any two positive integers a, n. Assume that $n \equiv (b_{l-1}, b_{l-2}, ..., b_0)$ is the binary representation of n where b_0 is the least significant bit. Note that $l = \texttt{len}(n)$. What is its

```
1: Initialize p ← 1,   m ← a,   i ← 0
2: repeat
3:     if b_i == 1 then
4:         p ← p * m
5:     end if
6:     m ← m^2,   i ← i + 1
7: until i == l
8: return p
```

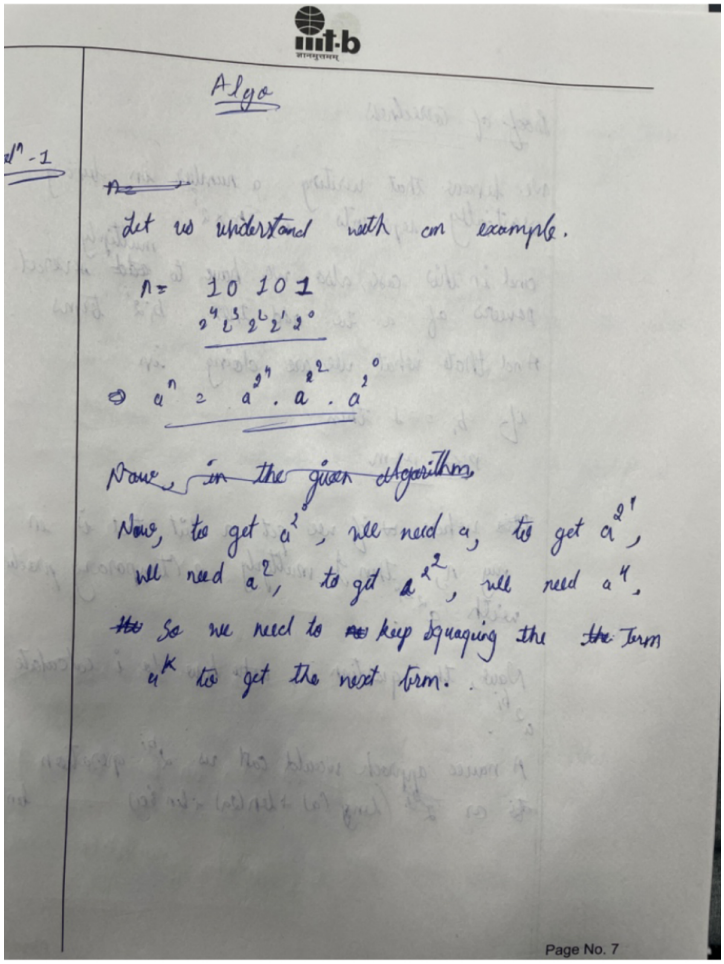
running time complexity in terms of l and $\texttt{len}(a)$, where $\texttt{len}(a)$ is the size of the representation (binary or otherwise) of a?

Uploaded File Details

Original Response

Previous Page: 1 / 4 Next

40 %



Previous Page: 1 / 4 Next

Evaluator Comments

About the complexity, the loop iterates for len(n) times. With every 1 in "n", the magnitude of p becomes close to a^n. The cost of multiplication is O(len(a^n)^2). In total, O(l* len(a^n)^2).

2 File Upload ✔ 8.0 Hide Answer

Application 1 of the Result of the Repeated Squaring Algorithm: Pseudo-random numbers are often generated using an algorithm called a *linear congruential generator*. In this we choose a relatively large modulus M (with unknown factorization), a multiplier a, a constant c and seed value X_0. Successive pseudo-random numbers are generated using the recurrence

$$X_n = a.X_{n-1} + c \mod M$$

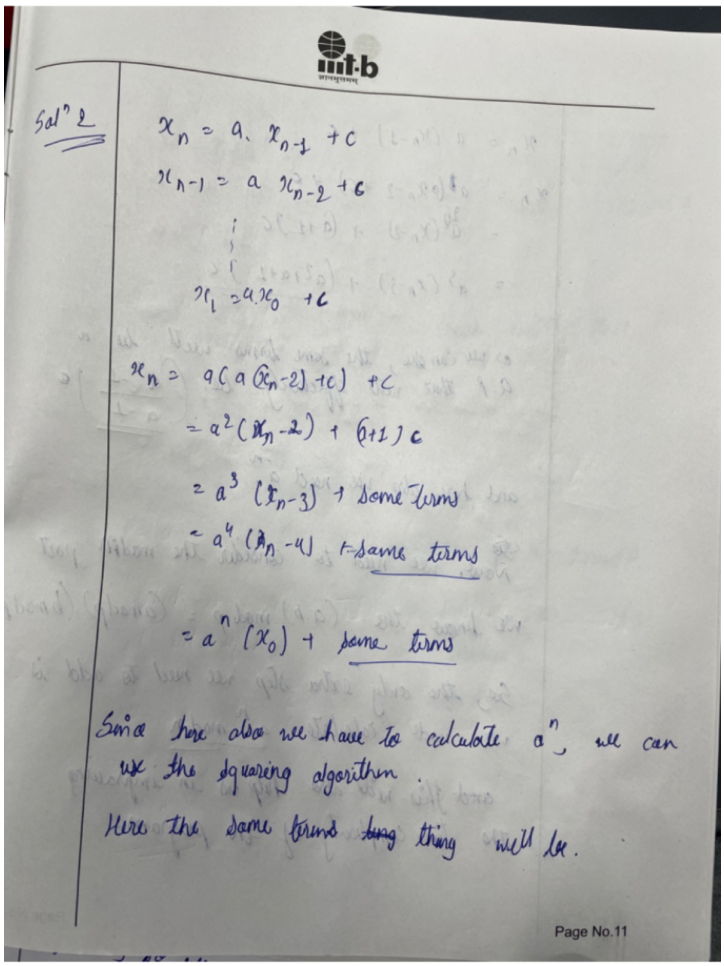
Give an algorithm to compute X_n in time polynomial in $\texttt{len}(n), \texttt{len}(M)$ assuming that $0 < a, c < M$.

Uploaded File Details

Original Response

Previous Page: 1 / 2 Next

40 %



Previous Page: 1 / 2 Next

3 File Upload ✔ 9.0 Hide Answer

Consider the following algorithm ($\texttt{len}(n)$ denotes the size of the representation – not necessarily binary – of n):

1: Initialize
$$k \leftarrow \left\lfloor \frac{\texttt{len}(n) - 1}{2} \right\rfloor, \quad m \leftarrow 2^k$$
2: **for** $i = (k-1)$ **downto** 0 **do**
3: **if** $(m + 2^i)^2 \leq n$ **then**
4: $m \leftarrow m + 2^i$
5: **end if**
6: **end for**
7: **return** m

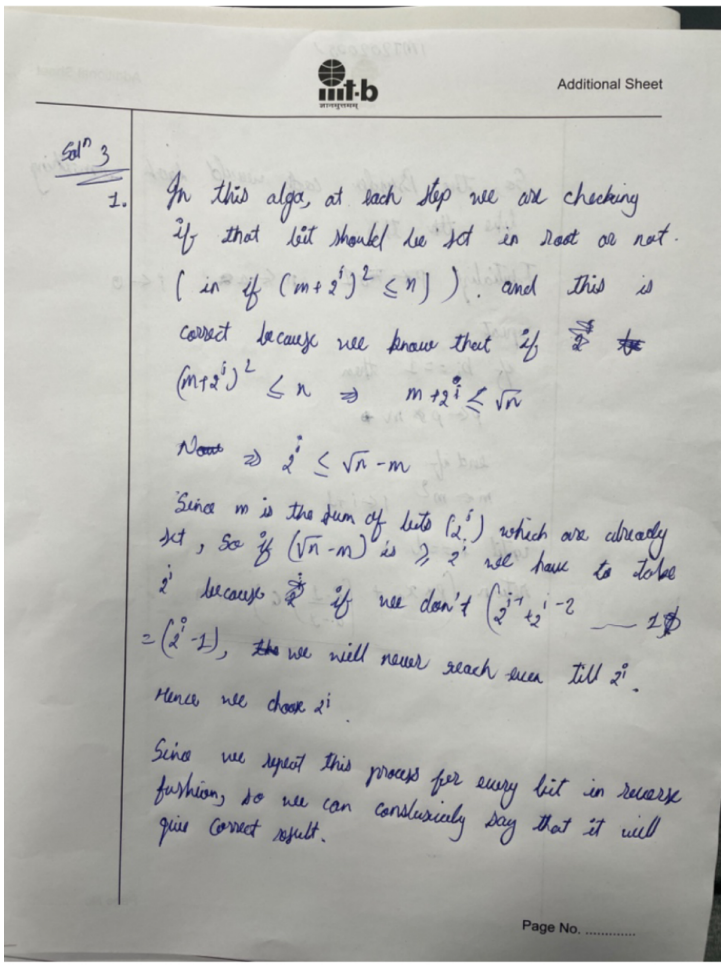
1. Show that this algorithm correctly computes $\lfloor \sqrt{n} \rfloor$. **Hint**: Think of the bit representation of m even though our internal representation may not be binary!!
2. Show how this algorithm can be implemented in time $O(\texttt{len}(n)^2)$. Can this be improved if we assume that we are working with a binary representation?
3. Extend this algorithm to compute $\lfloor n^{1/e} \rfloor$, assuming $n \geq 2^e$. What will its running time complexity be?

Uploaded File Details

Original Response

Previous Page: 1 / 3 Next

40 %



Additional Sheet

Soln 3

1. In this algo, at each step we are checking if that bit should be set in root or not. (ie if $(m+2^i)^2 \le n$)). and this is correct because we know that if $(m+2^i)^2 \le n \Rightarrow m+2^i \le \sqrt{n}$

Now $\Rightarrow 2^i \le \sqrt{n} - m$

Since m is the sum of bits (2^i) which are already set, So if $(\sqrt{n}-m)$ is $\ge 2^i$ we have to take 2^i because if we don't $(2^{i-1}+2^{i-2}+\dots+1)$ $= (2^i - 1)$, we will never reach even till 2^i.

Hence we chose 2^i.

Since we repeat this process for every bit in recursive fashion, so we can conclusively say that it will give correct result.

Page No. .....

Previous Page: 1 / 3 Next