

Tasks: Basic Number Theory 1

TIC TAC TOE

King TLE4EVER of time limit exceeded is really fascinated about Tic Tac Toe. He organizes a national level contest for Tic Tac Toe every year in time limit exceeded. every year the contest has lots of participants. This year there are n participants from all over the country. The king wants to know the sum of $w_i z_i$ from 1 to n . Now as you already know Moron is not good with maths he asks you to help him.

Aim: To write and execute the program for given scenario based on basic number theory-1

Algorithm:

1. Read the integer n .
2. calculate the minimum value of the sum $w_i (n-i) \pmod{10^9+7}$
3. calculate the maximum value of the sum $w_i (n-i) \pmod{10^9+7}$
4. Print the minimum and maximum value of the sum w_i output.
- end the program.

Case: Basic number theory

Output:

No. of inputs: 2

5 20 30

For the first case, the output is 1. For the second case, the output is 2. For the third case, the output is 3. For the fourth case, the output is 4. For the fifth case, the output is 5. For the sixth case, the output is 6. For the seventh case, the output is 7. For the eighth case, the output is 8. For the ninth case, the output is 9. For the tenth case, the output is 10.

Time: To compile and execute the program for given

1 - program code on basic number theory

Algorithm:

1. Read the input n.

2. Calculate the minimum value of the sum of the two numbers.

3. Output the result.

4. Calculate the maximum value of the sum of the two numbers.

5. Output the result.

6. Print the minimum and maximum value of the sum.

Output:

2. For the program

Program:

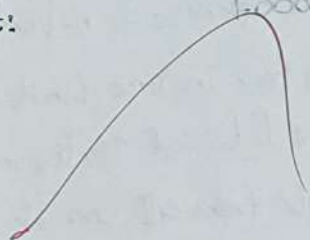
```
#include <stdio.h>
#include <math.h>
#define MOD 1000000007

int main()
{
    int t;
    scanf("%d", &t);
    while(t-- > 0)
    {
        long long n;
        scanf("%lld", &n);
        long long min_sum = ((n-1)*(n-1)*(n/4) % MOD);
        long long max_sum = ((n*(n*(n+1)*(3*(n+1)/(12)) % MOD);
        printf("%lld %lld\n", min_sum, max_sum);
    }
    return 0;
}
```

Problem:

Given two integers, and a recursive technique to find their GCD is the Euclidean Algorithm. The algorithm states that, for computing the GCD of two positive integers if and are equal there are few optimizations that can be made to the above logic.

1


$$\begin{aligned} \text{dom } f^{-1}(C_1) &= \{(x,y) \in X \times Y : (x,y) \in C_1\} \\ \text{dom } f^{-1}(C_2) &= \{(x,y) \in X \times Y : (x,y) \in C_2\} \\ \text{dom } f^{-1}(C_1 \cup C_2) &= \{(x,y) \in X \times Y : (x,y) \in C_1 \cup C_2\} \\ \text{dom } f^{-1}(C_1) \cup \text{dom } f^{-1}(C_2) &= \{(x,y) \in X \times Y : (x,y) \in C_1 \cup C_2\} \end{aligned}$$

for the above logic. We expect there are few optimisations that can be made that for computing the GCD of two positive integers is a better algorithm. The algorithm states given two integers, and a recursive technique to find their GCD is the Euclidean algorithm.

Algorithm:

1. Read the two positive integers a and b
2. If b is zero, return a as the GCD
3. Otherwise, recursively call the function with arguments b and the remainder of a divided by b .
4. Return the result of the recursive call as the GCD.

Program:

```
#include <stdio.h>
int gcd(int a, int b) {
    if (b == 0) {
        return a;
    }
    return gcd(b, a % b);
}
int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    int result = gcd(a, b);
    printf("%d\n", result);
    return 0;
}
```

Result: Thus the program is executed and verified successfully

VEL TECH - CSE	
NAME	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	7
RECORD (4)	7
TOTAL (15)	27
SIGN WITH DATE	