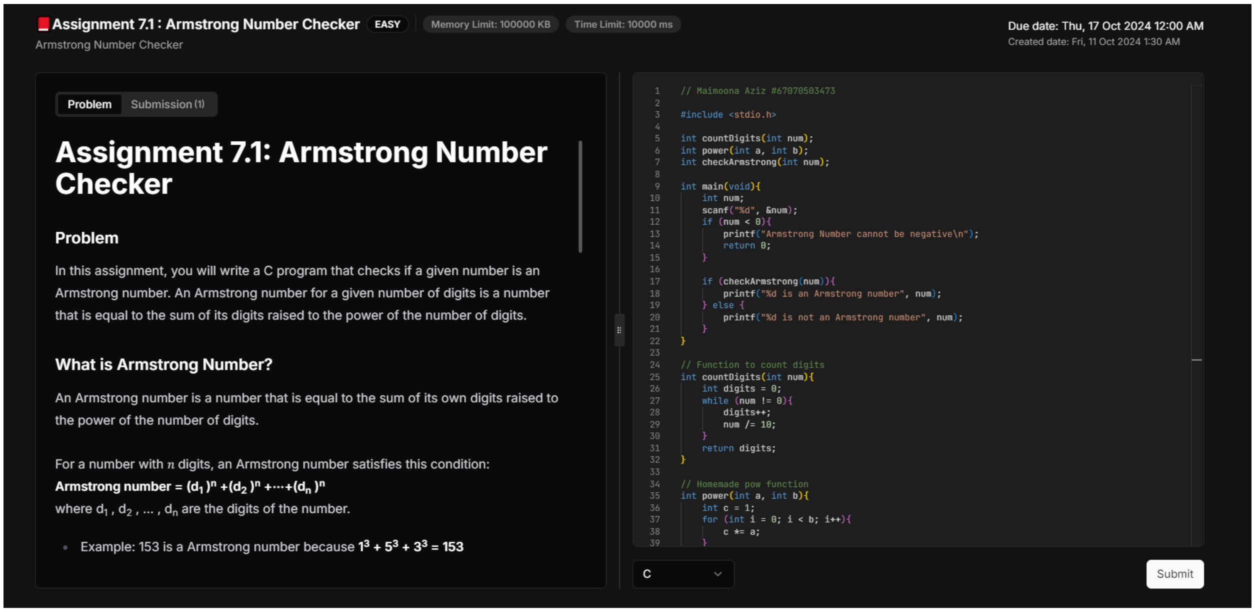


Assignment 7.1 : Armstrong Number Checker EASY Memory Limit: 100000 KB Time Limit: 10000 ms

Armstrong Number Checker

Due date: Thu, 17 Oct 2024 12:00 AM
Created date: Fri, 11 Oct 2024 1:30 AM

Assignment 7.1: Armstrong Number Checker

Problem

In this assignment, you will write a C program that checks if a given number is an Armstrong number. An Armstrong number for a given number of digits is a number that is equal to the sum of its digits raised to the power of the number of digits.

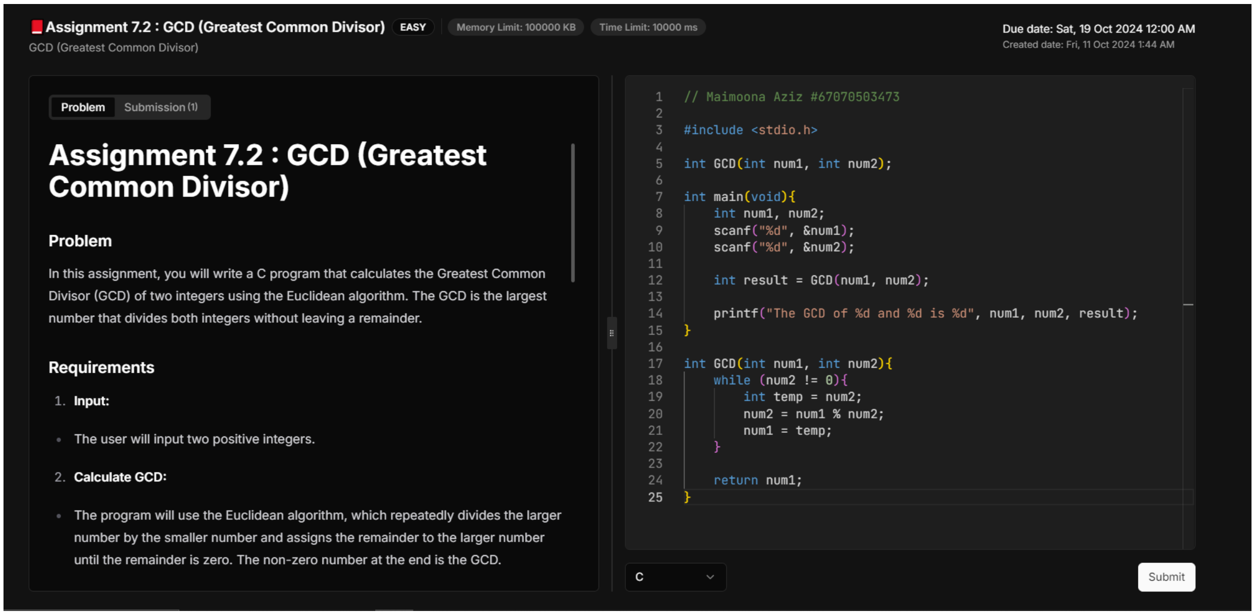
What is Armstrong Number?

An Armstrong number is a number that is equal to the sum of its own digits raised to the power of the number of digits.

For a number with n digits, an Armstrong number satisfies this condition:
Armstrong number = $(d_1)^n + (d_2)^n + \cdots + (d_n)^n$
where $d_1, d_2, \ldots, d_n$ are the digits of the number.

- Example: 153 is a Armstrong number because **$1^3 + 5^3 + 3^3 = 153$**

```c
// Maimoona Aziz #67070503473

#include <stdio.h>

int countDigits(int num);
int power(int a, int b);
int checkArmstrong(int num);

int main(void){
    int num;
    scanf("%d", &num);
    if (num < 0){
        printf("Armstrong Number cannot be negative\n");
        return 0;
    }

    if (checkArmstrong(num)){
        printf("%d is an Armstrong number", num);
    } else {
        printf("%d is not an Armstrong number", num);
    }
}

// Function to count digits
int countDigits(int num){
    int digits = 0;
    while (num != 0){
        digits++;
        num /= 10;
    }
    return digits;
}

// Homemade pow function
int power(int a, int b){
    int c = 1;
    for (int i = 0; i < b; i++){
        c *= a;
    }
```



Assignment 7.2 : GCD (Greatest Common Divisor) EASY Memory Limit: 100000 KB Time Limit: 10000 ms

GCD (Greatest Common Divisor)

Due date: Sat, 19 Oct 2024 12:00 AM
Created date: Fri, 11 Oct 2024 1:44 AM

Assignment 7.2 : GCD (Greatest Common Divisor)

Problem

In this assignment, you will write a C program that calculates the Greatest Common Divisor (GCD) of two integers using the Euclidean algorithm. The GCD is the largest number that divides both integers without leaving a remainder.

Requirements

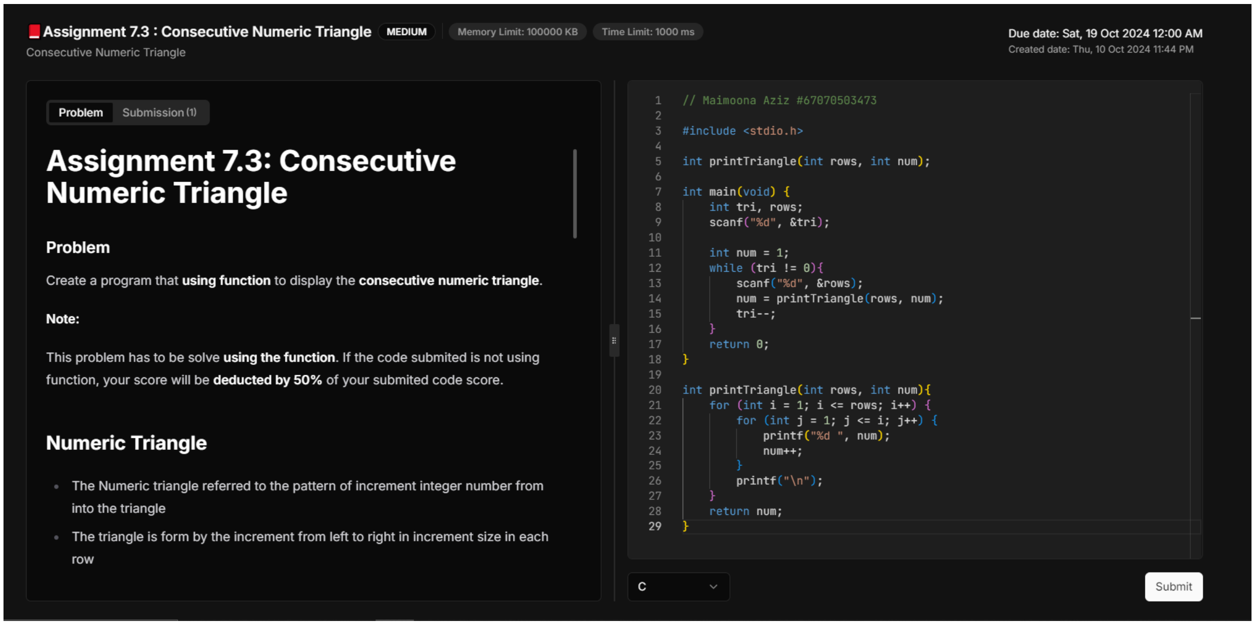
1. **Input:**

- The user will input two positive integers.

2. **Calculate GCD:**

- The program will use the Euclidean algorithm, which repeatedly divides the larger number by the smaller number and assigns the remainder to the larger number until the remainder is zero. The non-zero number at the end is the GCD.

```c
// Maimoona Aziz #67070503473

#include <stdio.h>

int GCD(int num1, int num2);

int main(void){
    int num1, num2;
    scanf("%d", &num1);
    scanf("%d", &num2);

    int result = GCD(num1, num2);

    printf("The GCD of %d and %d is %d", num1, num2, result);
}

int GCD(int num1, int num2){
    while (num2 != 0){
        int temp = num2;
        num2 = num1 % num2;
        num1 = temp;
    }

    return num1;
}
```



Assignment 7.3 : Consecutive Numeric Triangle MEDIUM Memory Limit: 100000 KB Time Limit: 1000 ms

Consecutive Numeric Triangle

Due date: Sat, 19 Oct 2024 12:00 AM
Created date: Thu, 10 Oct 2024 11:44 PM

Assignment 7.3: Consecutive Numeric Triangle

Problem

Create a program that **using function** to display the **consecutive numeric triangle**.

Note:

This problem has to be solve **using the function**. If the code submited is not using function, your score will be **deducted by 50%** of your submited code score.

Numeric Triangle

- The Numeric triangle referred to the pattern of increment integer number from into the triangle
- The triangle is form by the increment from left to right in increment size in each row

```c
// Maimoona Aziz #67070503473

#include <stdio.h>

int printTriangle(int rows, int num);

int main(void) {
    int tri, rows;
    scanf("%d", &tri);

    int num = 1;
    while (tri != 0){
        scanf("%d", &rows);
        num = printTriangle(rows, num);
        tri--;
    }
    return 0;
}

int printTriangle(int rows, int num){
    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d ", num);
            num++;
        }
        printf("\n");
    }
    return num;
}
```