TCS NQT 9th April 2025 Coding questions

Learning With Ram

Q1)Write a program that takes a single input string of space-separated integers, where:

The first two numbers indicate the number of rows (m) and columns (n) of a matrix.
The rest of the numbers represent the matrix elements (ages as non-negative integers).

Conditions to handle:

If the number of elements provided is more than m × n, print "Wrong input".
Otherwise, construct the matrix and check if each row has at least one prime number.

If all rows contain at least one prime → print "Valid".
If any row does not contain a prime → print "Not Valid".

✅ **Example:**
Suppose the input is:

2 2 3 4 5 6

row -2

col-2

matrix

3 4

5 6

numbers = [2, 2, 3, 4, 5, 6]
m = 2, n = 2, so we need a 2×2 matrix
We want:

matrix[0][0] = 3
matrix[0][1] = 4
matrix[1][0] = 5
matrix[1][1] = 6

matrix will be

3 4

5 6

Output:
Valid

2 2 4 6 8 10

2

2

4 6

8 10

invalid

Code:-

```
#include <iostream>
#include <sstream>
#include <vector>
using namespace std;

// Function to check if a number is prime
bool isPrime(int num) {
    if (num < 2) return false;
    for (int i = 2; i * i <= num; i++)
        if (num % i == 0)
            return false;
    return true;
}

int main() {
    string inputLine;
    getline(cin, inputLine);  // Read the full line of input

    stringstream ss(inputLine);
    vector<int> numbers;
    int num;
```

```
    while (ss >> num)
        numbers.push_back(num);

    // Check if there are at least two numbers (m and n)
    if (numbers.size() < 2) {
        cout << "Wrong input\n";
        return 0;
    }

    int m = numbers[0]; //row
    int n = numbers[1];//col
    int expectedSize = m * n;

    // Check if number of data elements exceeds matrix size
    if ((int)numbers.size() - 2 > expectedSize) {
        cout << "Wrong input\n";
        return 0;
    }

    // Fill the matrix
    vector<vector<int>> matrix(m, vector<int>(n, 0));
    int dataIndex = 2;
    for (int i = 0; i < m && dataIndex < numbers.size(); i++) {
        for (int j = 0; j < n && dataIndex < numbers.size(); j++) {
            /* This line assigns a value from the numbers vector to the matrix at position [i][j], and then
increments dataIndex by 1. */
            matrix[i][j] = numbers[dataIndex++];
        }
    }

    // Check if each row has at least one prime number
```

```
    bool allRowsValid = true;
    for (int i = 0; i < m; i++) {
        bool hasPrime = false;
        for (int j = 0; j < n; j++) {
            if (isPrime(matrix[i][j])) {
                hasPrime = true;
                break;
            }
        }
        if (!hasPrime) {
            allRowsValid = false;
            break;
        }
    }

    cout << (allRowsValid ? "Valid" : "Not Valid") << endl;

    return 0;
}
```

Q2)

Write a program to perform the following:
Take two positive integers m and n as input.
Find the m-th prime number and the n-th prime number.
For each of these two prime numbers, calculate the sum of its digits repeatedly until the result is a single-digit number (i.e., less than 10).

Let these results be m1 and n1.
Finally, print the value of m * m1.

eg. For input m = 5 and n = 6:

prime number sequences - 2 3 5 7 11 13
5th prime = 11 → digit sum = 1 + 1 = 2 → m1 = 2
6th prime = 13 → digit sum = 1 + 3 = 4 → n1 = 4

Final answer → m * m1 = 5 * 2 = 10

Code:

```
#include <iostream>
using namespace std;

// Function to check if a number is prime
bool isPrime(int num) {
   if (num < 2) return false;
   for (int i = 2; i * i <= num; i++)
      if (num % i == 0)
         return false;
   return true;
}

// Function to find the nth prime number
int findNthPrime(int n) {
   int count = 0, num = 2;
   while (true) {
      if (isPrime(num)) {
         count++;
         if (count == n)
            return num;
      }
      num++;
   }
}

// Function to reduce a number to a single-digit sum
int reduceToSingleDigit(int num) {
   while (num >= 10) {
```

```
        int sum = 0;
        while (num > 0) {
            sum += num % 10;
            num /= 10;
        }

        num = sum;
    }
    return num;
}

int main() {
    int m, n;
    cout << "Enter m and n: ";
    cin >> m >> n;

    int mthPrime = findNthPrime(m);
    int nthPrime = findNthPrime(n);

    int m1 = reduceToSingleDigit(mthPrime);
    int n1 = reduceToSingleDigit(nthPrime);
    cout << "Output: " << m * m1 << endl;

    return 0;
}
```

Please Support and subscribe our youtube channel for getting such informative content