#1 Multiplication table from 1 to 10

int Multiplication(int num1, int num2)

{

    return num1 \* num2;

}

void PrintTableHeader(int numOfMultipliers)

{

    cout << "\n\n\t\t\tMultiplication Table From1 to 10\n\n";

    for (int i = 1; i <= numOfMultipliers; i++)

    {

        cout << "\t" << i;

    }

    cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

}

string ColumnSeperator(int i)

{

    if (i == 10)

    {

        return "  |";

    }

    else

    {

        return "   |";

    }

}

void MultiplicationTable(int numOfMultipliers)

{

    PrintTableHeader(numOfMultipliers);

    for (int i = 1; i <= numOfMultipliers; i++)

    {

        cout << " " << i << ColumnSeperator(i) << "\t";

        for (int j = 1; j <= numOfMultipliers; j++)

        {

            cout << Multiplication(i, j) << "\t";

        }

        cout << endl;

    }

}

int main()

{

    MultiplicationTable(10);

}

#2 Print All prime Numbers From 1 to N

enum enPrimeNotPrime

{

    Prime = 1,

    NotPrime = 2

};

int ReadPositiveNumbers(string message)

{

    int num;

    do

    {

        cout << message << endl;

        cin >> num;

    } while (num <= 0);

    return num;

}

enPrimeNotPrime checkPrime(int number)

{

    int middleOfNum = round(number / 2);

    for (int counter = 2; counter <= middleOfNum; counter++)

    {

        if (number % counter == 0)

        {

            return enPrimeNotPrime ::NotPrime;

        }

    }

    return enPrimeNotPrime::Prime;

}

void PrintPrimeNumbersFrom1ToN(int N)

{

    cout << "\nOutput is :\n";

    for (int i = 1; i <= N; i++)

    {

        if (checkPrime(i) == enPrimeNotPrime::Prime)

        {

            cout << i << "\n";

        }

    }

}

int main()

{

    PrintPrimeNumbersFrom1ToN(ReadPositiveNumbers("please enter a positive number"));

}

#3 Perfect Number

Perfect number =sum (all divisors)

int ReadPositiveNumbers(string message)

{

    int num;

    do

    {

        cout << message << endl;

        cin >> num;

    } while (num <= 0);

    return num;

}

bool isPerfectnumber(int number)

{

    int middle = number / 2;

    int sumOfDivisors = 0;

    for (int i = 1; i <= middle; i++)

    {

        if (number % i == 0)

            sumOfDivisors += i;

    }

    return sumOfDivisors == number;

}

void printResult(int number)

{

    bool isPerfect = isPerfectnumber(number);

    if (isPerfect)

        cout << number << " is Perfect number ";

    else

        cout << number << " is not perfect number ";

}

int main()

{

    int number = ReadPositiveNumbers("Enter a positive number ");

    printResult(number);

}

#4 perfect number from 1 to n

int ReadPositiveNumbers(string message)

{

    int num;

    do

    {

        cout << message << endl;

        cin >> num;

    } while (num <= 0);

    return num;

}

bool isPerfectnumber(int number)

{

    int middle = number / 2;

    int sumOfDivisors = 0;

    for (int i = 1; i <= middle; i++)

    {

        if (number % i == 0)

            sumOfDivisors += i;

    }

    return sumOfDivisors == number;

}

void printPerfectNumbersFrom1toN(int N)

{

    cout << "Perfect numbers From 1 to " << N << " are :\n";

    for (int i = 1; i <= N; i++)

    {

        if (isPerfectnumber(i))

            cout << i << endl;

    }

}

int main()

{

    int N = ReadPositiveNumbers("Enter a positive number ");

    printPerfectNumbersFrom1toN(N);

}

#5 Print Digits in a Reversed Order

int ReadPositiveNumbers(string message)

{

    int num;

    do

    {

        cout << message << endl;

        cin >> num;

    } while (num <= 0);

    return num;

}

void printReversedDigits(int number)

{

    int reminder = 0;

    do

    {

        reminder = number % 10;

        cout << reminder << endl;

        number = number / 10;

    } while (number > 0);

}

int main()

{

    printReversedDigits(ReadPositiveNumbers("Enter a positive number :"));

}

#6 Sum of Digits

int ReadPositiveNumbers(string message)

{

    int num;

    do

    {

        cout << message << endl;

        cin >> num;

    } while (num <= 0);

    return num;

}

int SumOfDigits(int number)

{

    int reminder = 0, sum = 0;

    do

    {

        reminder = number % 10;

        sum += reminder;

        number = number / 10;

    } while (number > 0);

    return sum;

}

int main()

{

    int sumOfDigits = SumOfDigits(ReadPositiveNumbers("Enter a positive number :"));

    cout << "Sum OF Digits is : " << sumOfDigits;

}

#7 Reverse Number

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

int DivideTwoNumbers(int dividend, int divisor)

{

    return (dividend / divisor);

}

int ModulusTwoNumbers(int dividend, int divisor)

{

    return (dividend % divisor);

}

int ReversedOfNumber(int number)

{

    int modulus = 0;

    int revversed = 0;

    while (number != 0)

    {

        modulus = ModulusTwoNumbers(number, 10);

        revversed = (revversed \* 10) + modulus;

        number = DivideTwoNumbers(number, 10);

    }

    return revversed;

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    int n = ReversedOfNumber(number);

    cout << "reversed of  " << number << " is equal " << n;

}

#8 Digit frequency

using namespace std;

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

int CountDigitFrequency(short digit, int number)

{

    // index  على ال forloop لانه رقم ما بتقدري تعملي

    int freqCount = 0, reminder = 0;

    while (number != 0)

    {

        reminder = number % 10; //extract the last digit of the nnumber

        number = number / 10; //remove the last digit from the number

        if (reminder == digit)

        {

            freqCount++;

        }

    }

    return freqCount;

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    short digitToCheck = ReadPositiveNumber("enter  one digit to check");

    cout << "\n Digit  " << digitToCheck << " Frequency is  " << CountDigitFrequency(digitToCheck, number) << " Times(s) . \n";

}

#9 Print All Digits Frequencies of the number

#include <iostream>

#include <string>

using namespace std;

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

int CountDigitFrequency(short digit, int number)

{

    // index  على ال forloop لانه رقم ما بتقدري تعملي

    int freqCount = 0, reminder = 0;

    while (number != 0)

    {

        reminder = number % 10; // extract the last digit of the nnumber

        number = number / 10;   // remove the last digit from the number

        if (reminder == digit)

        {

            freqCount++;

        }

    }

    return freqCount;

}

void PrintAllDigitsFrequencies(int number)

{

    for (int i = 0; i < 10; i++)

    {

        int frequency = CountDigitFrequency(i, number);

        if (frequency > 0)

        {

            cout << "\n Digit  " << i << " Frequency is  " << frequency << " Times(s) . \n";

        }

    }

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    PrintAllDigitsFrequencies(number);

}

**# 10 print digits in order**

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

int DivideTwoNumbers(int dividend, int divisor)

{

    return (dividend / divisor);

}

int ModulusTwoNumbers(int dividend, int divisor)

{

    return (dividend % divisor);

}

int ReversedOfNumber(int number)

{

    int modulus = 0;

    int revversed = 0;

    while (number != 0)

    {

        modulus = ModulusTwoNumbers(number, 10);

        revversed = (revversed \* 10) + modulus;

        number = DivideTwoNumbers(number, 10);

    }

    return revversed;

}

void PrintDigits(int number)

{

    while (number > 0)

    {

        int reminder = ModulusTwoNumbers(number, 10);

        number = DivideTwoNumbers(number, 10);

        cout << reminder << endl;

    }

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    int n = ReversedOfNumber(number);

    PrintDigits(n);

}

**#11 palindrome number**

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

int DivideTwoNumbers(int dividend, int divisor)

{

    return (dividend / divisor);

}

int ModulusTwoNumbers(int dividend, int divisor)

{

    return (dividend % divisor);

}

int ReversedOfNumber(int number)

{

    int modulus = 0;

    int revversed = 0;

    while (number != 0)

    {

        modulus = ModulusTwoNumbers(number, 10);

        revversed = (revversed \* 10) + modulus;

        number = DivideTwoNumbers(number, 10);

    }

    return revversed;

}

bool isPalindrome(int number)

{

    return number == ReversedOfNumber(number);

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    bool isPalindrom = isPalindrome(number);

    if (isPalindrom)

    {

        cout << "Yes , " << number << " Is Palindrome number";

    }

    else

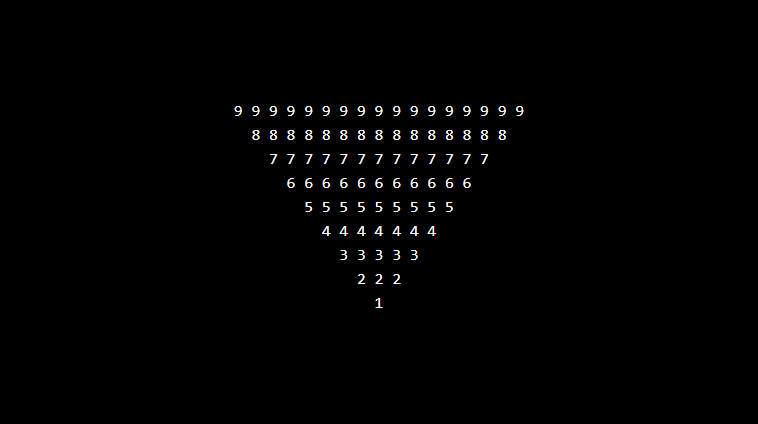
    {

        cout << "No , " << number << " Isn't Palindrome number";

    }

}

**#12 Inverted number pattern**

****

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

void PrintInvertedNumber(int number)

{

    for (int row = number; row >= 1; row--)

    {

        for (int column = 0; column < row; column++)

        {

            cout << row << " ";

        }

        cout << endl;

    }

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    PrintInvertedNumber(number);

}

**#13 Normal Number Pattern**

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

void PrintNumberPattern(int number)

{

    for (int row = 1; row <= number; row++)

    {

        for (int column = 0; column < row; column++)

        {

            cout << row << " ";

        }

        cout << endl;

    }

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    PrintNumberPattern(number);

}

**#14 Inverted letter Pattern**

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

void PrintInvertedLetterPatern(int number)

{

    for (int row = 65 + number -1 ; row >= 65; row--)

    {

        for (int column =1 ; column  <= number - (65+number -1 -row); column++)

        {

            cout << char(row) << " ";

        }

        cout << endl;

    }

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    PrintInvertedLetterPatern(number);

}

**#20 Normal letter Pattern**

int ReadPositiveNumber(string message)

{

    int number;

    do

    {

        cout << message << endl;

        cin >> number;

    } while (number < 0);

    return number;

}

void PrintInvertedLetterPatern(int number)

{

    for (int row = 65; row <= 65 + number - 1; row++)

    {

        for (int column = 1; column <= row - 65 + 1; column++)

        {

            cout << char(row) << " ";

        }

        cout << endl;

    }

}

int main()

{

    int number = ReadPositiveNumber("please enter positive number");

    PrintInvertedLetterPatern(number);

}