using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace IntroToCSharp

{

class Loops

{

public static void Exerxises6()

{

//////1./////////////////

Console.WriteLine("Please enter a number n and we will print integers from 1 to n");

int n = Convert.ToInt32(Console.ReadLine());

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

{

if (i == n) Console.Write(i + "\n");

else Console.Write(i + " ");

}

//////2./////////////////

Console.WriteLine("Now please input a number N and we will print the numbers from 1 to N that are not divisible by both 3 and 7 ");

int N = Convert.ToInt32(Console.ReadLine()), count2 = 1;

bool found = false;

do

{

if ((count2 % 3 == 0) && (count2 % 7 == 0)) { }

else

{

if (count2 == N) Console.Write(count2 + "\n");

else Console.Write(count2 + " ");

found = true;

}

count2 += 1;

}

while (count2 <= N);

if (found == false) Console.WriteLine("No numbers satisfying both requirements found");

//////3./////////////////

Console.WriteLine("Now enter up to 5 integers and we will print the smallest and largest of them");

int maxEntered = Int32.MaxValue, minEntered = Int32.MinValue, count3;

string I = Console.ReadLine();

count3 = I.ToString().Length;

int[] nums = new int[count3];

for (int iter = 0; iter < count3; iter++)

{

nums[iter] = Int32.Parse(I[iter].ToString());

}

maxEntered = nums.Max();

minEntered = nums.Min();

Console.WriteLine("The maximum number entered is: " + maxEntered + " and the minimum: " + minEntered);

//////4./////////////////

Console.WriteLine("Now we will print all of the combinations of a standard deck of cards");

string cardPak = "", cardNum = "";

for (int suite = 1; suite < 5; suite++)

{

switch (suite)

{

case 1:

cardPak = " of Spades";

break;

case 2:

cardPak = " of Hearts";

break;

case 3:

cardPak = " of Diamonds";

break;

case 4:

cardPak = " of Clubs";

break;

default:

break;

}

for (int card = 1; card < 13; card++)

{

switch (card)

{

case 1:

cardNum = "Ace";

break;

case 11:

cardNum = "Queen";

break;

case 12:

cardNum = "King";

break;

default:

break;

}

if ((card != 1) && (card != 11) && (card != 12))

{

Console.WriteLine(card + " " + cardPak);

}

else

{

Console.WriteLine(cardNum + " " + cardPak);

}

}

}

//////5./////////////////

Console.WriteLine("Now we will print the the sum of the first M letters of the Fibonacci sequence. Enter the value for M:");

int M = Int32.Parse(Console.ReadLine());

ulong nextNum = 0, currentNum = 0, lastNum = 0;

for (int count = 0; count < M; count++)

{

if (count < 2)

{

currentNum = lastNum = 1;

}

else

{

nextNum = currentNum + lastNum;

lastNum = currentNum;

currentNum = nextNum;

}

}

Console.WriteLine(currentNum);

//////6./////////////////

Console.WriteLine("Now we will write a program to calculate N!/K! for selected N and K where (1<K<N)");

Console.WriteLine("Please enter value for K and then N:");

int K = Convert.ToInt32(Console.ReadLine());

ulong Kfactorial = 1, Nfactorial = 1;

int N6 = Convert.ToInt32(Console.ReadLine());

if (N > K)

{

for (ulong power = 1; power < (ulong)K; power++)

{

Kfactorial \*= power;

}

for (ulong power = 1; power < (ulong)N6; power++)

{

Nfactorial \*= power;

}

Console.WriteLine("{0}", (Nfactorial / Kfactorial));

}

else Console.WriteLine("The values selected are invalid");

////////7./////////////////

Console.WriteLine("Now we will write a program to calculate N!\*K!/(N-K)! for selected N and K where (1<K<N)");

Console.WriteLine("Please enter value for K and then N:");

int K7 = Convert.ToInt32(Console.ReadLine());

ulong Kfactorial7 = 1, Nfactorial7 = 1, NminusKFactorial = 1;

int N7 = Convert.ToInt32(Console.ReadLine()), NminusK = 1;

if (N7 > K7)

{

NminusK = N7 - K7;

for (int power = 1; power < NminusK; power++)

{

NminusKFactorial \*= (ulong)power;

}

for (ulong power = 1; power < (ulong)K7; power++)

{

Kfactorial7 \*= power;

}

for (ulong power = 1; power < (ulong)N7; power++)

{

Nfactorial7 \*= power;

}

Console.WriteLine("\n{0}", (Nfactorial \* Kfactorial) / NminusKFactorial);

}

else Console.WriteLine("The values selected are invalid");

////////8./////////////////

Console.WriteLine("We will now calculate the p'th Catalan number based on a given value of p in the Catalan formula (2p)!/(p+1)!p! Please enter a value for p: ");

int p = Convert.ToInt32(Console.ReadLine());

ulong pTimes2Factorial = 1, pPlusOneFactorial = 1, pFactorial = 1;

for (int iter = 1; iter < 2 \* p; iter++)

{

pTimes2Factorial \*= (ulong)iter;

}

for (int iter = 1; iter < p + 1; iter++)

{

pPlusOneFactorial \*= (ulong)iter;

}

for (int iter = 1; iter < p; iter++)

{

pFactorial \*= (ulong)iter;

}

Console.WriteLine("\n{0}", pTimes2Factorial / (pPlusOneFactorial \* pFactorial));

//////9./////////////////

Console.WriteLine("Given j and x, we will now calculate the sum of a series S = 1 + 1!/x + 2!/x^2, 3!/x^3 + j!/x^j...Please enter a value for j and then x:");

int j = Convert.ToInt32(Console.ReadLine()), x = Convert.ToInt32(Console.ReadLine());

double jfactorial = 1, S = 0;

for (int iter = 0; iter < j; iter++)

{

jfactorial = 1;

for (int rep = 1; rep <= iter; rep++)

{

jfactorial \*= (double)rep;

}

S += jfactorial / Math.Pow(x, iter);

}

Console.WriteLine("\n{0}", S);

////////10.////////////////

Console.WriteLine("We will print a matrix based on the value M entered. Please Enter M: ");

int B = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("\n");

for (int o = 1; o <= B; o++)

{

for (int y = o; y <= o + (B - 1); y++)

{

Console.Write(y + " ");

}

Console.WriteLine("\n");

}

////////11.////////////////

Console.WriteLine("Now we will see how many zeroes there are in the factorial of the integer inputted. Enter value: ");

int R = Convert.ToInt32(Console.ReadLine());

ulong factorial11 = 1, numOfZeroes = 0;

for (int u = 1; u <= R; u++)

{

factorial11 \*= (ulong)u;

}

if (factorial11 % 10 == 0)

{

do

{

factorial11 /= 10;

numOfZeroes += 1;

}

while (factorial11 % 10 == 0);

Console.WriteLine("There are " + numOfZeroes + " zeroes in the factorial of the given integer\r\n");

}

//////12.////////////////

Console.WriteLine("Now we will convert a given integer into its binary representation. Enter a value: ");

int T = Convert.ToInt32(Console.ReadLine()), slider = T;

for (int bit = 7; bit >= 0; bit--)

{

if ((T - Math.Pow(2, bit)) >= 0)

{

if (slider - (int)Math.Pow(2, bit) >= 0)

{

Console.Write("1");

slider -= (int)Math.Pow(2, bit);

}

else Console.Write("0");

}

else Console.Write("0");

if (bit == 0) Console.Write("\n");

}

////////13.////////////////

Console.WriteLine("Now we will convert a given binary string into its decimal representation. Enter a 8-bit binary value: ");

string binaryIn = Console.ReadLine();

int decimalEquiv = 0, shift = 7;

if (binaryIn.Length == 8)

{

for (int bit = 0; bit <= 7; bit++)

{

if (Convert.ToInt32(binaryIn[bit]) == 49)

{

decimalEquiv += (int)Math.Pow(2, shift);

}

shift--;

}

Console.WriteLine("The decimal equivalent of " + binaryIn + " is: " + decimalEquiv);

}

else

{

Console.WriteLine("Invalid binary Value entered");

}

////////14.////////////////

Console.WriteLine("Now we will convert a given integer into its hexadecimal representation. Enter a value: ");

int K14 = Convert.ToInt32(Console.ReadLine()), sliderHex = K14, highestDiv = 0;

string highestDivStr = "";

Console.Write("The hexadecimal representation of " + K14 + " is: 0x");

for (int pow = 3; pow >= 0; pow--)

{

if (Math.Pow(16, pow) < K14)

{

highestDiv = sliderHex / (int)Math.Pow(16, pow);

sliderHex -= (highestDiv \* (int)Math.Pow(16, pow));

switch (highestDiv)

{

case 10:

highestDivStr = "A";

break;

case 11:

highestDivStr = "B";

break;

case 12:

highestDivStr = "C";

break;

case 13:

highestDivStr = "D";

break;

case 14:

highestDivStr = "E";

break;

case 15:

highestDivStr = "F";

break;

default:

break;

}

if (highestDiv <= 9)

{

Console.Write(highestDiv);

}

else Console.Write(highestDivStr);

}

else Console.Write("0");

}

Console.Write("\r\n");

////////15.////////////////

Console.WriteLine("Now we will convert a hexadecimal value into its decimal representation. Enter a value i.e: 0x0001");

double sliderDec = 0;

int powerOf16 = 2, overNine = 0;

string P = Console.ReadLine();

bool flagLetter = false;

for (int pow = 3; pow >= 0; pow--)

{

switch (P[powerOf16])

{

case 'A':

overNine = 10;

flagLetter = true;

break;

case 'B':

overNine = 11;

flagLetter = true;

break;

case 'C':

overNine = 12;

flagLetter = true;

break;

case 'D':

overNine = 13;

flagLetter = true;

break;

case 'E':

overNine = 14;

flagLetter = true;

break;

case 'F':

overNine = 15;

flagLetter = true;

break;

default:

break;

}

double check = Char.GetNumericValue(P[powerOf16]);

if (flagLetter == true)

{

sliderDec += overNine \* (int)Math.Pow(16, pow);

}

else sliderDec += check \* Math.Pow(16, pow);

flagLetter = false;

powerOf16++;

}

Console.Write("The decimal equivalent of " + P + " is: " + sliderDec + "\n");

////////16.////////////////

Console.WriteLine("Enter a value and we will print the values from 1 to this value N in a random order: ");

int RandN = Convert.ToInt32(Console.ReadLine()), N16 = 0, counter = 0;

bool oneSpotted = false, NSpotted = false;

do

{

byte[] set = new byte[RandN + 200];

Random randoms = new Random();

randoms.NextBytes(set);

for (int ctr = 0; ctr < set.Length; ctr++)

{

if (set[ctr] == RandN)

{

N16 = set[ctr];

NSpotted = true;

break;

}

}

if (NSpotted == true)

{

for (int ctr = 0; ctr < set.Length; ctr++)

{

if (set[ctr] == 1)

{

Console.Write(set[ctr] + " ");

counter = 1;

oneSpotted = true;

}

}

for (int ctr = 0; ctr < set.Length; ctr++)

{

if (oneSpotted == false) Console.Write("1 ");

if ((set[ctr] != 1) & ((set[ctr] <= RandN)))

{

Console.Write(set[ctr] + " ");

counter++;

if ((counter) % 10 == 0) Console.Write("\n");

}

if (ctr + 1 == set.Length)

{

Console.Write(N16 + "\n\n");

}

}

}

}

while (NSpotted == false);

////////17.////////////////

Console.WriteLine("Now enter two numbers S17 and K17 and we will determine what is the greatest common denominator between them as well as the least common multiple: ");

int S17 = Convert.ToInt32(Console.ReadLine()), K17 = Convert.ToInt32(Console.ReadLine()), denominator = 2, resM = 0, resK = 0, GCD = 0;

do

{

if ((S17 % denominator == 0) & (K17 % denominator == 0))

{

GCD = denominator;

}

denominator++;

if (denominator == Math.Max(S17, K17)) Console.WriteLine("The GCM of " + S17 + " and " + K17 + " is: " + GCD + " and the LCM is: {0}", Math.Abs(S17 \* K17) / GCD);

}

while (denominator < Math.Max(S17, K17));

////////18.////////////////

Console.WriteLine("Now we will present a two-dimensional matrix assembled in a spiral given the dimensions W x W. Enter W: ");

int W = Convert.ToInt32(Console.ReadLine()), WCount = W, cellValue = 0, dirChange = 0, cellColumn = 0, cellRow;

int[,] matrix = new int[W, W];

for (int square = 0; square < W; square++)

{

//right

matrix[0, square] = square + 1;

cellValue = square + 2;

cellColumn = square;

cellRow = 0;

if (square + 1 == W)

{

dirChange++;

do

{

WCount--;

//down

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

{

matrix[++cellRow, cellColumn] = cellValue++;

//cellRow = row;

}

//left

dirChange++;

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

{

matrix[cellRow, --cellColumn] = cellValue++;

}

WCount--;

//up

dirChange++;

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

{

matrix[--cellRow, cellColumn] = cellValue++;

}

//right

dirChange++;

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

{

matrix[cellRow, ++cellColumn] = cellValue++;

}

dirChange++;

}

while(dirChange < (2 \* W) - 1);

}

}

Console.WriteLine();

for (int row = 0; row < matrix.GetLength(0); row++)

{

for (int col = 0; col < matrix.GetLength(1); col++)

{

Console.Write(" " + matrix[row, col]);

}

Console.WriteLine();

}

}

}

}