**06\_ LOOPS**

**Pr.01\_Number from 1 to N**

Write a program that enters from the console a positive integer **n** and **prints all the numbers from 1 to n**, on a single line, separated by a space. Examples:

|  |  |
| --- | --- |
| **n** | **output** |
| 3 | 1 2 3 |
| 5 | 1 2 3 4 5 |

using System;

class NumbersFrom1ToN

{

static void Main()

{

Console.Write("Enter a number n = ");

int n = int.Parse(Console.ReadLine());

if (n < 1)

{

Console.WriteLine("Invalid number.");

}

else

{

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

{

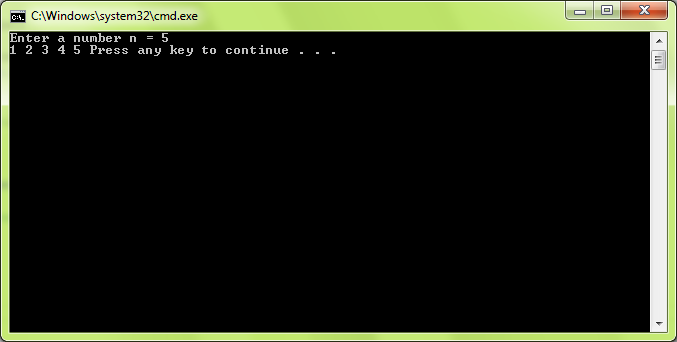
Console.Write(i + " ");

}

}

}

}



**Pr.02\_Numbers Not Divisible By 3 and 7**

Write a program that enters from the console a positive integer **n** and prints all the **numbers from 1 to n not divisible by 3 and 7**, on a single line, separated by a space. Examples:

|  |  |
| --- | --- |
| **n** | **output** |
| 3 | 1 2 |
| 10 | 1 2 4 5 8 10 |

using System;

class NumbersNotDivisibleBy3and7

{

static void Main()

{

Console.Write("Enter a number n = ");

int n = int.Parse(Console.ReadLine());

if (n < 1)

{

Console.WriteLine("Invalid number.");

}

else

{

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

{

if (i % 3 == 0 || i % 7 == 0)

{

continue;

}

Console.Write(i + " ");

}

}

}

}



**Pr.03\_Min Max Sum Average Of Num**

Write a program that reads from the console a sequence of **n** integer numbers and returns the **minimal**, the **maximal** number, the sum and the average of all numbers (displayed with 2 digits after the decimal point). The **input** starts by the number **n** (alone in a line) followed by **n lines**, each holding an integer number. The **output** is like in the examples below. Examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **input** | **output** |  | **input** | **output** |
| 3  2  5  1 | min = 1  max = 5  sum = 8  avg = 2.67 | 2  -1  4 | min = -1  max = 4  sum = 3  avg = 1.50 |

using System;

class MinMaxSumAverageOfNum

{

static void Main()

{

Console.Write("Enter a number n = ");

int n = int.Parse(Console.ReadLine());

int max = int.MinValue;

int min = int.MaxValue;

double sum = 0;

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

{

int num = int.Parse(Console.ReadLine());

if (num > max)

{

max = num;

}

if (num < min)

{

min = num;

}

sum += num;

}

Console.WriteLine("min = " + min);

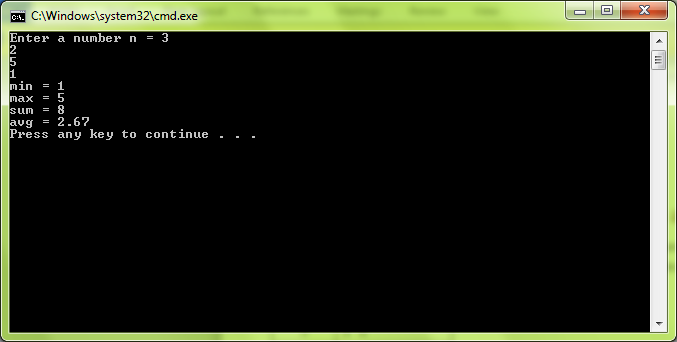
Console.WriteLine("max = " + max);

Console.WriteLine("sum = " + sum);

Console.WriteLine("avg = {0:F2} ", (sum/n));

}

}



**Pr.04\_PrintingDeckOfCards**

Write a program that generates and prints **all possible cards from a** [**standard deck of 52 cards**](http://en.wikipedia.org/wiki/Standard_52-card_deck) (without the jokers). The cards should be printed using the classical notation (like 5♠, A♥, 9♣ and K♦). The card faces should start from 2 to A. Print each card face in its four possible suits: clubs, diamonds, hearts and spades. Use 2 nested for-loops and a switch-case statement.

|  |
| --- |
| **output** |
| 2♣ 2♦ 2♥ 2♠  3♣ 3♦ 3♥ 3♠  …  K♣ K♦ K♥ K♠  A♣ A♦ A♥ A♠ |

using System;

class PrintingDeckOfCards

{

static void Main()

{

string[] cards = { "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K", "A" };

char spades = '\u2660';

char hearts = '\u2665';

char diams = '\u2666';

char clubs = '\u2663';

for (int i = 0; i < cards.Length; i++)

{

for (int j = 1; j <= 4; j++)

{

switch (j)

{

case 1:

Console.Write(cards [i] + clubs + " ");

break;

case 2:

Console.Write(cards [i] + diams + " ");

break;

case 3:

Console.Write(cards[i] + hearts + " ");

break;

case 4:

Console.Write(cards[i] + spades + " ");

break;

}

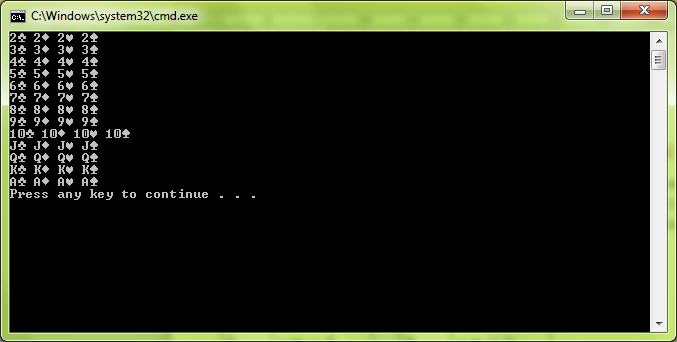
}

Console.WriteLine();

}

}

}



## Pr.05\_Calculate 1 + 1!/X + 2!/X2 + … + N!/XN

Write a program that, for a given two integer numbers **n** and **x**, calculates the sum S = 1 + 1!/x + 2!/x2 + … + n!/xn. Use only one loop. Print the result with 5 digits after the decimal point.

|  |  |  |
| --- | --- | --- |
| **n** | **x** | **S** |
| 3 | 2 | 2.75000 |
| 4 | 3 | 2.07407 |
| 5 | -4 | 0.75781 |

using System;

class CalculatingSumN\_X

{

static void Main()

{

Console.Write("Enter n = ");

int n = int.Parse(Console.ReadLine());

Console.Write("Enter x = ");

int x = int.Parse(Console.ReadLine());

double product = 1;

double index = 1;

double sum = 1;

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

{

product \*= i;

index \*= x;

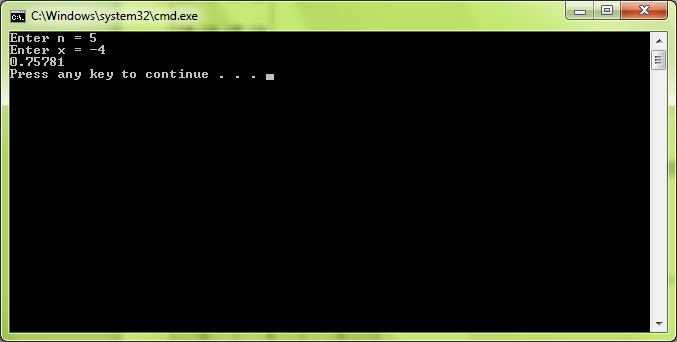
sum += (product / index);

}

Console.WriteLine("{0:F5}", sum);

}

}



**Pr.06\_CalculatingN!/ K!**

Write a program that calculates **n! / k!** for given **n** and **k** (1 < **k** < **n** < 100). Use only one loop. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **k** | **n! / k!** |
| 5 | 2 | 60 |
| 6 | 5 | 6 |
| 8 | 3 | 6720 |

using System;

class CalculatingNKfactorial

{

static void Main()

{

Console.Write("Enter a number n, (1 < n < 100): n = ");

int n = int.Parse(Console.ReadLine());

Console.Write("Enter a number k, (1 < k < n < 100), k = ");

int k = int.Parse(Console.ReadLine());

double nFactorial = 1;

double kFactorial = 1;

if (k < 1 || n < 1 || k > 100 || n > 100 || k > n)

{

Console.WriteLine("Invalid number.");

}

else

{

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

{

nFactorial \*= i;

if (i == k)

{

kFactorial = nFactorial;

}

}

Console.WriteLine(nFactorial / kFactorial);

}

}

}



**Pr.07\_Calculating N! / (K! \* (N-K)!)**

In combinatorics, the number of ways to choose **k** different members out of a group of **n** different elements (also known as the number of [**combinations**](http://en.wikipedia.org/wiki/Combination)) is calculated by the following formula:

 \binom nk = \frac{n!}{k!(n-k)!},

For example, there are 2598960 ways to withdraw 5 cards out of a standard deck of 52 cards. Your task is to write a program that calculates **n! / (k! \* (n-k)!)** for given **n** and **k** (1 < **k** < **n** < 100). Try to use only two loops. Examples:

|  |  |  |
| --- | --- | --- |
| **n** | **k** | **n! / (k! \* (n-k)!)** |
| 3 | 2 | 3 |
| 4 | 2 | 6 |
| 10 | 6 | 210 |
| 52 | 5 | 2598960 |

using System;

class CalculatingNumberOfCombinations

{

static void Main()

{

Console.Write("Enter a number n, (1 < n < 100): n = ");

int n = int.Parse(Console.ReadLine());

Console.Write("Enter a number k, (1 < k < n < 100), k = ");

int k = int.Parse(Console.ReadLine());

int nMinusk = n - k;

double nFactorial = 1;

double kFactorial = 1;

double nMinusKfcatorial = 1;

if (k < 1 || n < 1 || k > 100 || n > 100 || k > n)

{

Console.WriteLine("Invalid number.");

}

else

{

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

{

nFactorial \*= i;

if (i == (n - k))

{

nMinusKfcatorial = nFactorial;

}

if (i == k)

{

kFactorial = nFactorial;

}

}

Console.WriteLine(nFactorial / (kFactorial \* nMinusKfcatorial));

}

}

}



**Pr.08\_Catalan Numbers**

In combinatorics, the [Catalan numbers](http://en.wikipedia.org/wiki/Catalan_number) are calculated by the following formula:

C_n = \frac{1}{n+1}{2n\choose n} = \frac{(2n)!}{(n+1)!\,n!} = \prod\limits_{k=2}^{n}\frac{n+k}{k} \qquad\mbox{ for }n\ge 0.

Write a program to calculate the **nth Catalan number** by given **n** (1 < n < 100). Examples:

|  |  |
| --- | --- |
| **n** | **Catalan(n)** |
| 0 | 1 |
| 5 | 42 |
| 10 | 16796 |
| 15 | 9694845 |

using System;

using System.Numerics;

class CatalanNumbers

{

static void Main()

{

Console.Write("Enter a number n, (1 < n < 100): n = ");

int n = int.Parse(Console.ReadLine());

BigInteger n2Factorial = 1;

BigInteger nPlus1Factorial = 1;

BigInteger nFactorial = 1;

BigInteger result;

if (n < 1 || n > 100)

{

Console.WriteLine("Invalid number.");

}

else

{

for (int i = 1; i <= 2 \* n; i++)

{

n2Factorial \*= i;

if (n == i)

{

nFactorial = n2Factorial;

}

if (n + 1 == i)

{

nPlus1Factorial = n2Factorial;

}

}

result = n2Factorial / (nPlus1Factorial \* nFactorial);

Console.WriteLine(result);

}

}

}



**Pr.09\_Matrix Of Numbers**

Write a program that reads from the console a positive integer number n (1 ≤ n ≤ 20) and **prints a matrix** like in the examples below. Use two nested loops. Examples:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | **matrix** |  | **n** | **matrix** |  | **n** | **matrix** |
| 2 | 1 2  2 3 | 3 | 1 2 3  2 3 4  3 4 5 | 4 | 1 2 3 4  2 3 4 5  3 4 5 6  4 5 6 7 |

using System;

class MatrixOfNumbers

{

static void Main()

{

Console.Write("Enter a number n, (1 <= n <= 20): n = ");

int n = int.Parse(Console.ReadLine());

if (n < 1 || n > 20)

{

Console.WriteLine("Invalid number.");

}

else

{

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

{

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

{

Console.Write(j + " ");

}

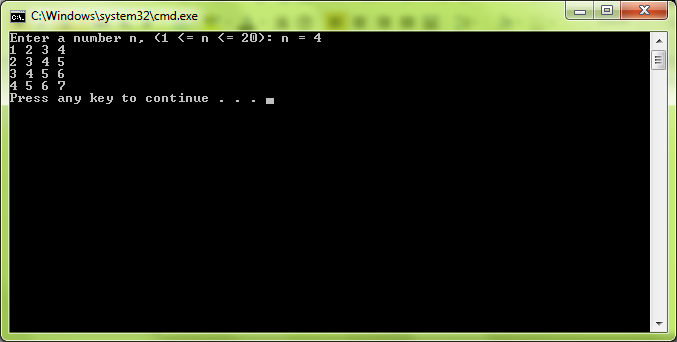
Console.WriteLine();

}

}

}

}



**Pr.10\_Odd And Even Product**

using System;

class OddAndEvenProduct

{

static void Main()

{

Console.Write("Enter n = ");

int n = int.Parse(Console.ReadLine());

int oddNums = 1;

int evenNums = 1;

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

{

int index = int.Parse(Console.ReadLine());

if (i % 2 == 0)

{

evenNums \*= index;

}

else

{

oddNums \*= index;

}

}

if (evenNums == oddNums)

{

Console.WriteLine("yes");

Console.WriteLine("product = " + evenNums);

}

else

{

Console.WriteLine("no");

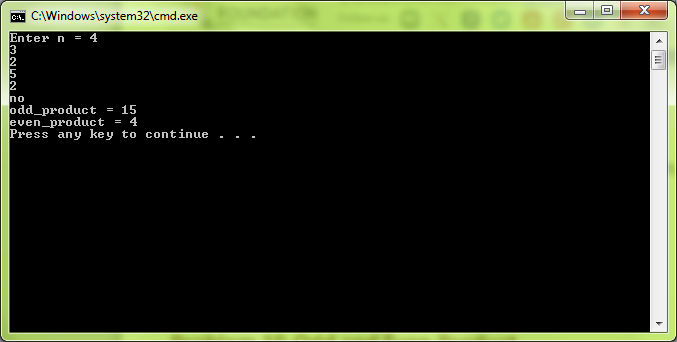
Console.WriteLine("odd\_product = " + oddNums);

Console.WriteLine("even\_product = " + evenNums);

}

}

}



**Pr.11\_Random Numbers In Given Range**

Write a program that enters 3 integers n, min and max (min ≤ max) and prints n random numbers in the range [min...max]. Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **min** | **max** | **random numbers** |
| 5 | 0 | 1 | 1 0 0 1 1 |
| 10 | 10 | 15 | 12 14 12 15 10 12 14 13 13 11 |

Note that the above output is just an example. Due to randomness, your program most probably will produce different results.

using System;

class RandomNumbersInGivenRange

{

static void Main()

{

int n = int.Parse(Console.ReadLine());

int min = int.Parse(Console.ReadLine());

int max = int.Parse(Console.ReadLine());

Random rnd = new Random();

int val;

for (int i = 0; i < n; i++)

{

Console.WriteLine(rnd.Next(min, max + 1));

}

}

}

