C# Chapter 10

1. Write a program to simulate n **nested loops** from 1 to n.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_10\_Solution\_1

{

class Program

{

static void Loops(int[] arr, int index)

{

if (index >= arr.Length)

{

foreach (int element in arr) Console.Write("{0} ", element);

Console.WriteLine();

}

else

for (int i = 1; i <= arr.Length; i++)

{

arr[index] = i;

Loops(arr, index + 1);

}

}

static void Main(string[] args)

{

Console.Write("Enter N: ");

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

int[] arr = new int[n];

Loops(arr, 0);

}

}

}

4. You are given a **set of strings**. Write a **recursive program**, which **generates all subsets**, consisting exactly k strings chosen among the elements of this set.  
Example input:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_10\_Solution\_4

{

class Program

{

static string[] wordsArr;

static void FindSubsets(int[] arr, int index, int start, int end)

{

if (index >= arr.Length)

{

Console.Write("(");

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

{

Console.Write("{0}", wordsArr[arr[i]]);

if (i != arr.Length - 1) Console.Write(" ");

}

Console.Write("), ");

}

else

for (int i = start; i < end; i++)

{

arr[index] = i;

FindSubsets(arr, index + 1, i + 1, end);

}

}

static void Main(string[] args)

{

Console.Write("Enter array length: ");

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

wordsArr = new string[length];

Console.WriteLine();

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

{

Console.Write("Enter {0} word: ", i + 1);

wordsArr[i] = Console.ReadLine();

}

Console.Write("\nEnter K: ");

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

int[] arr = new int[k];

Console.WriteLine();

FindSubsets(arr, 0, 0, length);

Console.ReadLine();

}

}

}

5.  Write a **recursive program**, which prints **all subsets of a given set of N** words.

using System;

namespace Chapter\_10\_Solution\_5

{

class Program

{

static string[] strings, str;

static int length;

static void cycle(int iter, int index, int k)

{

if (iter == k)

{

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

Console.WriteLine("({0})", str[i]);

return;

}

for (int i = index; i < strings.Length; i++)

{

str[iter] = strings[i];

cycle(iter + 1, i + 1, k);

}

}

static void Main(string[] args)

{

Console.Write("Enter words length: ");

length = Int32.Parse(Console.ReadLine());

strings = new string[length];

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

{

Console.Write("Enter {0} word: ", i + 1);

strings[i] = Console.ReadLine();

}

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

{

str = new string[length];

cycle(0, 0, i);

}

}

}

}

6. Implement the **merge-sort algorithm recursively**. In it the initial array is divided into two equal in size parts, which are sorted (recursively via merge-sort) and after that the two sorted parts are merged in order to get the whole sorted array.

using System;

namespace Chapter\_10\_Solution\_6

{

class Program

{

static public void DoMerge(int[] numbers, int left, int mid, int right)

{

int[] temp = new int[25];

int i, left\_end, num\_elements, tmp\_pos;

left\_end = (mid - 1);

tmp\_pos = left;

num\_elements = (right - left + 1);

while ((left <= left\_end) && (mid <= right))

{

if (numbers[left] <= numbers[mid])

temp[tmp\_pos++] = numbers[left++];

else

temp[tmp\_pos++] = numbers[mid++];

}

while (left <= left\_end)

temp[tmp\_pos++] = numbers[left++];

while (mid <= right)

temp[tmp\_pos++] = numbers[mid++];

for (i = 0; i < num\_elements; i++)

{

numbers[right] = temp[right];

right--;

}

}

static public void MergeSort\_Recursive(int[] numbers, int left, int right)

{

int mid;

if (right > left)

{

mid = (right + left) / 2;

MergeSort\_Recursive(numbers, left, mid);

MergeSort\_Recursive(numbers, (mid + 1), right);

DoMerge(numbers, left, (mid + 1), right);

}

}

static void Main(string[] args)

{

Console.Write("Enter array length: ");

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

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i + 1);

arr[i] = Int32.Parse(Console.ReadLine());

}

Console.WriteLine("\nSorted");

MergeSort\_Recursive(arr, 0, length - 1);

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

Console.WriteLine(arr[i]);

}

}

}

7. Write a recursive program, which generates and prints **all permutations of the numbers** 1, 2, …, n, for a given integer n.

using System;

namespace Chapter\_10\_Solution\_7

{

class Program

{

static void FindCombinations(int[] arr, int index)

{

if (index == arr.Length)

{

Console.Write("{");

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

if (i < arr.Length - 1) Console.Write("{0} ", arr[i]);

else Console.Write(arr[i]);

Console.Write("}, ");

}

else

{

for (int i = index; i < arr.Length; i++)

{

int temp = arr[i];

arr[i] = arr[index];

arr[index] = temp;

FindCombinations(arr, index + 1);

temp = arr[i];

arr[i] = arr[index];

arr[index] = temp;

}

}

}

static void Main(string[] args)

{

Console.Write("Enter N: ");

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

int[] arr = new int[n];

for (int i = 0; i < arr.Length; i++) arr[i] = i + 1;

FindCombinations(arr, 0);

}

}

}

8. You are given an array of integers and a number N. Write a recursive program that finds **all subsets** of numbers in the array, which have a **sum** N. For example, if we have the array {2, 3, 1, -1} and N=4, we can obtain N=4 as a sum in the following two ways: 4=2+3-1; 4=3+1.

using System;

namespace Chapter\_10\_Solution\_8

{

class Program

{

public static bool isSubsetSum(int[] arr, int n, int sum)

{

bool[,] subset = new bool[sum + 1, n + 1];

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

subset[0,i] = true;

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

subset[i,0] = false;

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

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

{

subset[i,j] = subset[i, j-1];

if(i >= arr[j - 1])

subset[i,j] = subset[i,j] || subset[i - arr[j-1], j-1];

}

return subset[sum, n];

}

static void Main(string[] args)

{

Console.Write("Enter arr length: ");

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

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i + 1);

arr[i] = Int32.Parse(Console.ReadLine());

}

Console.Write("Enter sum: ");

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

if (isSubsetSum(arr, arr.Length, sum) == true)

Console.WriteLine("Found a subset with given sum");

else

Console.WriteLine("No subset with given sum");

Console.ReadLine();

}

}

}

9. You are given an array of **positive** integers. Write a program that checks whether there is one or more numbers in the array **(subset), whose sum is equal to S**. Can you solve the task **efficiently for large arrays**?

using System;

namespace Chapter\_10\_Solution\_9

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter array length: ");

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

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i + 1);

arr[i] = Int32.Parse(Console.ReadLine());

}

Console.Write("Enter sum: ");

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

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

{

int first = arr[i];

for (int j = i + 1; j < arr.Length; j++)

{

int second = arr[j];

if ((first + second) == sum)

Console.WriteLine("({0}, {1}) ", first, second);

}

}

}

}

}

10. You are given a **matrix** with passable and impassable cells. Write a recursive program that finds **all paths between two cells** in the matrix.

using System;

namespace Chapter\_10\_Solution\_10

{

class Program

{

static int numberOfPaths(int m, int n)

{

int[,] count = new int[m,n];

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

count[i,0] = 1;

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

count[0,j] = 1;

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

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

count[i,j] = count[i-1,j] + count[i,j-1];

return count[m-1,n-1];

}

static void Main(string[] args)

{

Console.WriteLine("Possible paths: {0}", numberOfPaths(3, 3));

}

}

}