

Chapter 10. Recursion

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

```
using System;

namespace ex1
{
    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);
            Console.ReadKey();
        }
    }
}
```

C:\Users\LumiDaK1NG

Enter N: 3

1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3
1 3 1
1 3 2
1 3 3
2 1 1
2 1 2
2 1 3
2 2 1
2 2 2
2 2 3
2 3 1
2 3 2
2 3 3
3 1 1
3 1 2
3 1 3
3 2 1
3 2 2
3 2 3
3 3 1
3 3 2
3 3 3

—

2. Write a program to generate all variations with duplicates of n elements class k. Sample input:

n = 3

k = 2

Sample output

(1 1), (1 2), (1 3), (2 1), (2 2), (2 3), (3 1), (3 2), (3 3)

Think about and implement an iterative algorithm for the same task

using System;

namespace ex2

{

class Program

{

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

{

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 = start; i <= end; i++)

{

arr[index] = i;

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

}

}

static void Main(string[] args)

{

Console.Write("Enter N: ");

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

Console.Write("Enter K: ");

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

int[] arr = new int[k];

GetCombinations(arr, 0, 1, n);

Console.ReadKey();

} } }



```
C:\Users\LumiDaK1NG\Desktop\UNI\programim\semestri II\0
Enter N: 3
Enter K: 2
(1 1), (1 2), (1 3), (2 2), (2 3), (3 3),
```

3. Write a program to generate and print all combinations with duplicates of k elements from a set with n elements. Sample input:

n = 3

k = 2

Sample output:

(1 1), (1 2), (1 3), (2 2), (2 3), (3 3)

Think about and implement an iterative algorithm for the same task.

```
using System;

namespace ex3
{
    class Program
    {
        static void GetCombinations(int[] arr, int index, int start, int end)
        {
            if (index >= arr.Length)
            {
                Console.WriteLine("");
                for (int i = 0; i < arr.Length; i++)
                {
                    if (i < arr.Length - 1) Console.Write("{0} ", arr[i]);
                    else Console.Write(arr[i]);
                    Console.WriteLine(", ");
                }
            }
            else
            {
                for (int i = start; i <= end; i++)
                {
                    arr[index] = i;
                    GetCombinations(arr, index + 1, i, end);
                }
            }
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Enter N: ");
            int n = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Enter K: ");
            int k = Int32.Parse(Console.ReadLine());
            int[] arr = new int[k];
            GetCombinations(arr, 0, 1, n);
            Console.ReadKey(); } } }
```



Select C:\Users\LumiDaK1NG\Desktop\UNI\programim\src

Enter N: 3
Enter K: 2
(1 1), (1 2), (1 3), (2 2), (2 3), (3 3),

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. Sample input:

strings = {'test', 'rock', 'fun'}

k = 2

Sample output:

(test rock), (test fun), (rock fun)

Think about and implement an iterative algorithm as well.

```
using System;

namespace ex4
{
    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);
            }
        }

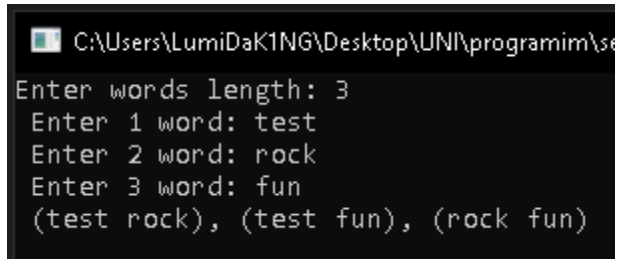
        static void Main(string[] args)
        {
            Console.Write("Enter words length: ");
            length = Int32.Parse(Console.ReadLine());

            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);  
        }  
        Console.ReadKey();  
    }  
}
```



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\LumiDaK1NG\Desktop\UNI\programim\se". The command prompt displays the following text:

```
Enter words length: 3  
Enter 1 word: test  
Enter 2 word: rock  
Enter 3 word: fun  
(test rock), (test fun), (rock fun)
```

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

words = {'test', 'rock', 'fun'}

Example output:

(), (test), (rock), (fun), (test rock), (test fun),

(rock fun), (test rock fun)

Think about and implement an iterative algorithm for the same task.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

```
namespace ex5
{
    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);
    }
    Console.ReadKey();
}
}
}

```

```

C:\Users\LumiDaK1NG\Desktop\
Enter words length: 3
Enter 1 word: test
Enter 2 word: rock
Enter 3 word: fun
(
(
(
(test )
(
(
(rock)
(
(
(fun)
(
(
(test )
(rock)
(
(test )
(fun)
(
(rock)
(fun)
(
(test )
(rock)
(fun)

```


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

n = 3

Example output:

(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)

Try to find an iterative solution for generating permutations.

```
using System;

namespace ex6
{
    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());

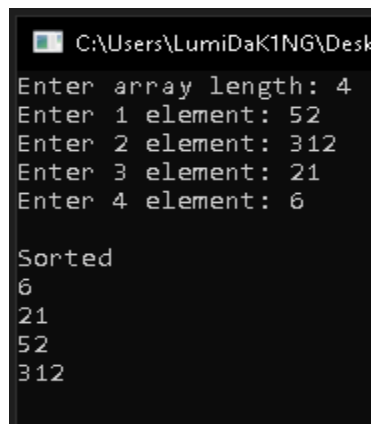
    int[] arr = new int[length];

    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]);
    Console.ReadKey();
}
}
}

```



```

C:\Users\LumiDaK1NG\Desktop>
Enter array length: 4
Enter 1 element: 52
Enter 2 element: 312
Enter 3 element: 21
Enter 4 element: 6

Sorted
6
21
52
312

```

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 ex8
{
    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];
        }

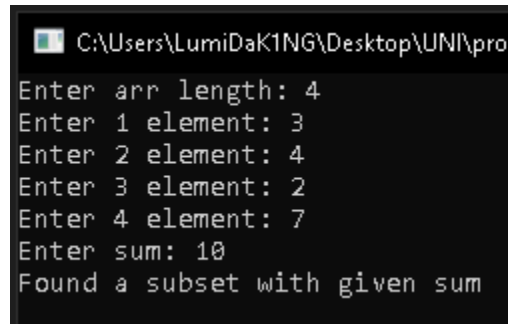
        static void Main(string[] args)
        {
            Console.Write("Enter arr length: ");
            int length = Int32.Parse(Console.ReadLine());
            int[] arr = new int[length];

            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 found with given sum");
        }
    }
}
```

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



```
C:\Users\LumiDaK1NG\Desktop\UNI\pro  
Enter arr length: 4  
Enter 1 element: 3  
Enter 2 element: 4  
Enter 3 element: 2  
Enter 4 element: 7  
Enter sum: 10  
Found a subset with given sum
```

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;

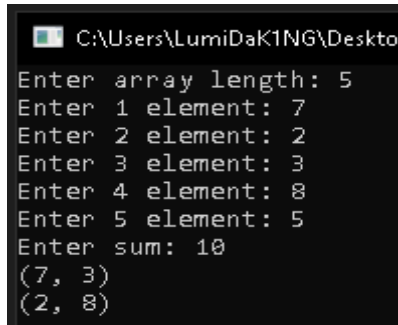
namespace ex9
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter array length: ");
            int length = Int32.Parse(Console.ReadLine());
            int[] arr = new int[length];

            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);
                }
            }
            Console.ReadKey();
        }
    }
}
```



```
C:\Users\LumiDaK1NG\Desktop
Enter array length: 5
Enter 1 element: 7
Enter 2 element: 2
Enter 3 element: 3
Enter 4 element: 8
Enter 5 element: 5
Enter sum: 10
(7, 3)
(2, 8)
```

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 ex10
{
    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(42, 312));
            Console.ReadKey();
        }
    }
}
```



A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\LumiDaK1NG\Desktop\UNI\p". The console output displays the text "Possible paths: 1114229696" followed by a cursor on a new line.