



# CST-250 Activity 3: Recursion Examples in C# Guide

## Contents

Overview .....	2
Example 1 – Count to One .....	2
Example 2 – Factorial .....	2
Example 3 – Greatest Common Divisor .....	4
Example 4 – Knight's Tour .....	6



## Overview

**Objective:** We will solve several problems to see examples of recursion in action.

## Background

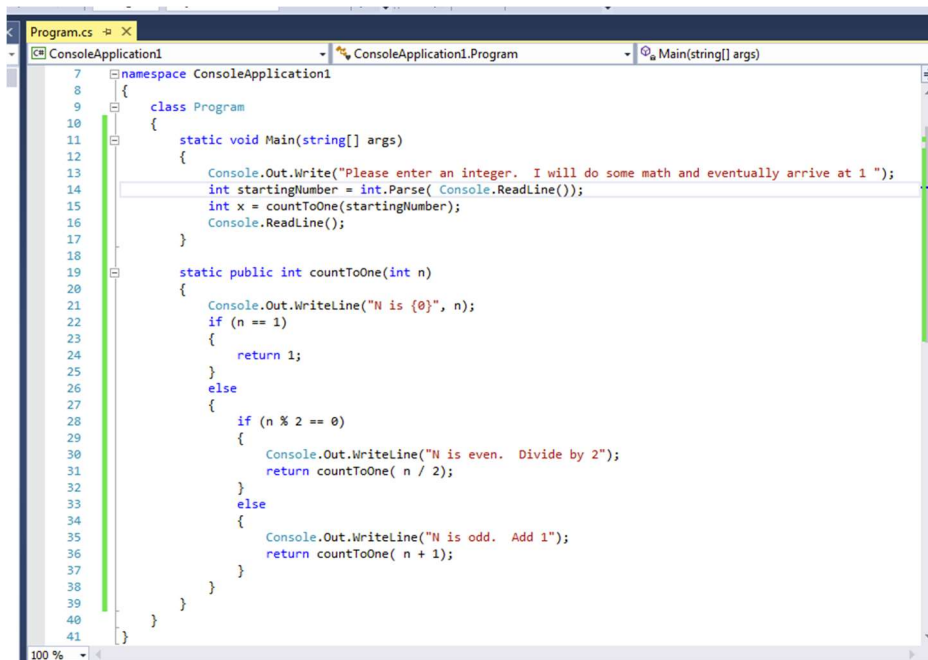
Recursion is an iterative way about and solving a problem where a function calls itself. Read "Recursion," located in the topic materials.

### Example 1 – Count to One

In this function, we start with any positive integer. For each iteration, do one of two things:

- If the number is 1, then stop.
- If the number is even, then divide by 2.
- If the number is odd, then add 1.

Type the following program and test it with a variety of numbers.



```
Program.cs
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Please enter an integer. I will do some math and eventually arrive at 1 ");
            int startingNumber = int.Parse(Console.ReadLine());
            int x = countToOne(startingNumber);
            Console.ReadLine();
        }

        static public int countToOne(int n)
        {
            Console.WriteLine("N is {0}", n);
            if (n == 1)
            {
                return 1;
            }
            else
            {
                if (n % 2 == 0)
                {
                    Console.WriteLine("N is even. Divide by 2");
                    return countToOne(n / 2);
                }
                else
                {
                    Console.WriteLine("N is odd. Add 1");
                    return countToOne(n + 1);
                }
            }
        }
    }
}
```

### Deliverables:

1. Take a screenshot of the application being run and put it into a Microsoft Word document. Caption the picture with a description of what is being demonstrated.
2. ZIP file containing the project folder.

### Example 2 – Factorial



Factorial is a formula that multiplies a number by itself and every integer less than itself. For example,

$$5! = (5 \times 4) \times 3 \times 2 \times 1 = (20 \times 3) \times 2 \times 1 = (60 \times 2) \times 1 = 120$$

We can approach for solving a factorial problem in at least in two standard ways:

First, we can use the standard "for loop" or iterative method.

$$4! = 1 \times 2 \times 3 \times 4 \text{ or } 4 \times 3 \times 2 \times 1 = 24$$

Secondly, we can use recursive thinking.

$$\begin{aligned} 4! &= 4 * 3! \\ &= 4 * (3 * 2!) \\ &= 4 * (3 * (2 * 1!)) \\ &= 4 * (3 * (2 * (1))) \\ &= 24 \end{aligned}$$

$$\text{Fact}(n) = \begin{cases} 1 & , \text{when } n = 0 \\ n * \text{fact}(n-1) & , \text{when } n > 0 \end{cases}$$

Here is an example of a recursive factorial program. Calculate the factorial value for a variety of numbers. See if there is a limit to the program.



```
Program.cs  Program.cs
Factorial  Factorial.Program
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Factorial
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             int startingNumber = 6;
14             Console.WriteLine(factorial(startingNumber));
15             Console.ReadLine();
16         }
17
18         static int factorial(int x)
19         {
20             Console.WriteLine("x is {0}", x);
21             if (x == 1)
22             {
23                 return 1;
24             }
25             else
26             {
27                 return x * factorial(x - 1);
28             }
29         }
30     }
31 }
32
```

### Deliverables:

1. Take a screenshot of the application being run and put it into a Microsoft Word document. Caption the picture with a description of what is being demonstrated.
2. ZIP file containing the project folder.

## Example 3 – Greatest Common Divisor

First, let's go back to middle school math for a refresher in terminology.

A **divisor** evenly divides another number. For example, 15 is divisible by 15, 5, 3, and 1.

$$15 / 15 = 1$$

$$15 / 5 = 3$$

$$15 / 3 = 5$$

$$15 / 1 = 15$$

A **common divisor** is a number that evenly divides two different numbers. For example, a common divisor for 20 and 30 is 5.



$$20 / 5 = 4$$

$$30 / 5 = 6$$

The **greatest common divisor** is the largest of all common divisors. For example, the greatest common divisor for 20 and 30 is 10.

$$20 / 10 = 2$$

$$30 / 10 = 3$$

There is a method for computing the greatest common divisor (GCD) for any two integers.

- 1) Start with two numbers: For example, 440 and 80.
- 2) Divide the first number by the second and record the remainder.
- 3) If the remainder is 0, then stop. The GCD answer is the last remainder found.
- 4) If the remainder is not zero, then repeat the process using the second number and the remainder in the next step.

N 1	N 2	N1 % N2 = remainder
440	80	440 / 80 = 5 remainder 40
80	40	80 / 40 = 0 remainder 40
40	40	40 / 40 = 1 remainder 0

The greatest common divisor is the last remainder that you found. In this case, the GCD is 40.

Now it is your turn. Calculate the GCD for these examples:

Find the GCD for 180 and 150.

N 1	N 2	N1 % N2 = remainder
180	150	

Find the GCD for 400 and 85.

N 1	N 2	N1 % N2 = remainder
400	85	




Now let's write a program to find the GCD for us:

```

Program.cs  X
GreatestCommonDivisor  GreatestCommonDivisor.Program  gc
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace GreatestCommonDivisor
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             int number1 = 400;
14             int number2 = 85;
15             int answer = gcd(number1, number2);
16             Console.WriteLine("The gcd of {0} and {1} is {2}", number1, number2, answer);
17             Console.ReadLine();
18         }
19
20         static int gcd(int n1, int n2)
21         {
22             if (n2 == 0)
23             {
24                 return n1;
25             }
26             else
27             {
28                 Console.WriteLine("Not yet. Remainder is {0}", n1 % n2);
29                 return gcd(n2, n1 % n2);
30             }
31         }
32     }
33 }

```

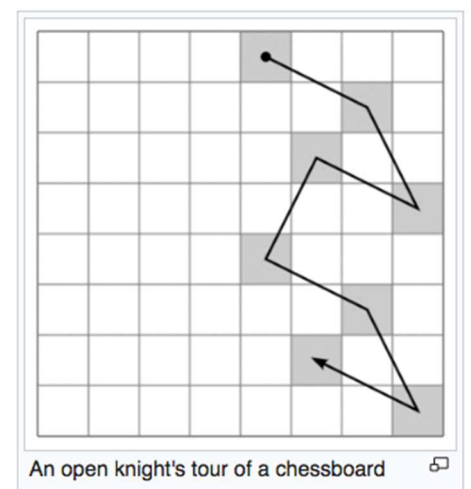
### Deliverables:

1. Take a screenshot of the application being run and put it into a Microsoft Word document. Caption the picture with a description of what is being demonstrated.
2. ZIP file containing the project folder.

## Example 4 – Knight's Tour

The Knight's Tour is a chessboard challenge to move a knight to all spaces on the board without visiting a square twice.

- 1) Place a knight in the corner of a chessboard.
- 2) Mark the starting square as "visited."
- 3) Move to another square and mark it.
- 4) Continue moving until you have visited all squares on the board exactly once.





The solution we will look at is a "brute force" solver. That is, the computer will attempt every combination of moves until he finds a solution or concludes that no solutions exist.

Here is the algorithm:

- 1) Visit square (0,0) and mark it as "visited." Theoretically, a piece can start on any square, but most squares require immense amounts of time to solve.
- 2) Choose the first of the possible legal moves.
- 3) If there are no legal moves, then move backward and recursively try the next legal move.
- 4) Repeat #3 until either a solution is found or until all possible moves have been attempted.

```
Program.cs
KnightsTour
KnightsTour.Program
solveKT()

1 using System;
2
3 namespace KnightsTour
4 {
5
6     class Program
7     {
8         static int BoardSize = 8;
9         static int attemptedMoves = 0;
10        /* xMove[] and yMove[] define next move of Knight.
11         xMove[] is for next value of x coordinate
12         yMove[] is for next value of y coordinate */
13        static int[] xMove = { 2, 1, -1, -2, -2, -1, 1, 2 };
14        static int[] yMove = { 1, 2, 2, 1, -1, -2, -2, -1 };
15
16        // boardGrid is an 8x8 array that contains -1 for an unvisited square or a move number between 0 and 63.
17        static int[,] boardGrid = new int[BoardSize, BoardSize];
18
19        // Driver Code
20        public static void Main()
21        {
22            solveKT();
23            Console.ReadLine();
24        }
25
26        /* This function solves the Knight Tour problem using backtracking. This function uses solveKTUtil() to
27        solve the problem. It returns false if no complete tour is possible, otherwise return true and prints the tour. Please note
28        that there may be more than one solution. */
29        static void solveKT()
30        {
31            /* Initialization of solution matrix. value of -1 means "not visited yet" */
32            for (int x = 0; x < BoardSize; x++)
33                for (int y = 0; y < BoardSize; y++)
34                    boardGrid[x, y] = -1;
35            int startX = 0;
36            int startY = 4;
37            // set starting position for knight
38            boardGrid[startX, startY] = 0;
39
40            // count the total number of guesses
41            attemptedMoves = 0;
42
43            /* explore all tours using solveKTUtil() */
44            if (!solveKTUtil(startX, startY, 1))
45            {
46                Console.WriteLine("Solution does not exist for {0} {1}", startX, startY);
47            }
48            else
49            {
50                printSolution(boardGrid);
51                Console.Out.WriteLine("Total attempted moves {0}", attemptedMoves);
52            }
53        }
54    }
```



```

55  /* A recursive utility function to solve Knight Tour problem */
56  static bool solveKTUtil(int x, int y, int moveCount)
57  {
58      attemptedMoves++;
59      if (attemptedMoves % 1000000 == 0) Console.WriteLine("Attempts: {0}", attemptedMoves);
60
61      int k, next_x, next_y;
62
63      // check to see if we have reached a solution. 64 = moveCount
64      if (moveCount == BoardSize * BoardSize)
65          return true;
66
67      /* Try all next moves from the current coordinate x, y */
68      for (k = 0; k < 8; k++)
69      {
70          next_x = x + xMove[k];
71          next_y = y + yMove[k];
72          if (isSquareSafe(next_x, next_y))
73          {
74              boardGrid[next_x, next_y] = moveCount;
75              if (solveKTUtil(next_x, next_y, moveCount + 1))
76                  return true;
77              else
78                  // backtracking
79                  boardGrid[next_x, next_y] = -1;
80          }
81      }
82      return false;
83  }
84
85  /* A utility function to check if i,j are valid indexes for N*N chessboard */
86  static bool isSquareSafe(int x, int y)
87  {
88      return (x >= 0 && x < BoardSize &&
89              y >= 0 && y < BoardSize &&
90              boardGrid[x, y] == -1);
91  }
92
93  /* A utility function to print solution matrix sol[N][N] */
94  static void printSolution(int[,] solution)
95  {
96      for (int x = 0; x < BoardSize; x++)
97      {
98          for (int y = 0; y < BoardSize; y++)
99              Console.Write(solution[x, y] + " ");
100
101          Console.WriteLine();
102      }
103  }
104
105  }
106
107  }
108
109

```

## Improvement:

The previous example shows a solution that is not very efficient. A computer scientist named Warnsdorff found a better solution. He said that when choosing the next move for the knight, choose the square that has the least number of unvisited squares.

Improve the previous program to implement Warnsdorff's solution. You will have to create a function named something like "int CountVisitedNeighbors()". Use this function to compare all of the knight's possible moves. Choose the next move where the most neighbors have already been visited.

See a demonstration of "Knight's Tour Warnsdorff's Rule" located in the topic materials.

The code used in this exercise is based on the following pages:

The Knight's Tour Problem | Backtracking-1 (n.d.) from the Geeks for Geeks website.

<https://www.geeksforgeeks.org/the-knights-tour-problem-backtracking-1/>

Warnsdorff's Algorithm for Knight's Tour Problem (n.d.) from the Geeks for Geeks website.

<https://www.geeksforgeeks.org/warnsdorffs-algorithm-knights-tour-problem/>





**Deliverables:**

1. Take a screenshot of the application being run and put it into a Microsoft Word document. Caption the picture with a description of what is being demonstrated.
2. ZIP file containing the project folder.

**SPECIAL NOTE:** There are four parts to this activity. Be sure to include results from all four.