CST-227 Activity 3: Recursion Examples in C# Guide

Contents

[Overview 2](#_Toc5009208)

[Example 1 – Count to One 2](#_Toc5009209)

[Example 2 – Factorial 3](#_Toc5009210)

[Example 3 – Greatest Common Divisor 4](#_Toc5009211)

[Example 4 – Knight's Tour 6](#_Toc5009212)

# Overview

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

**Background**

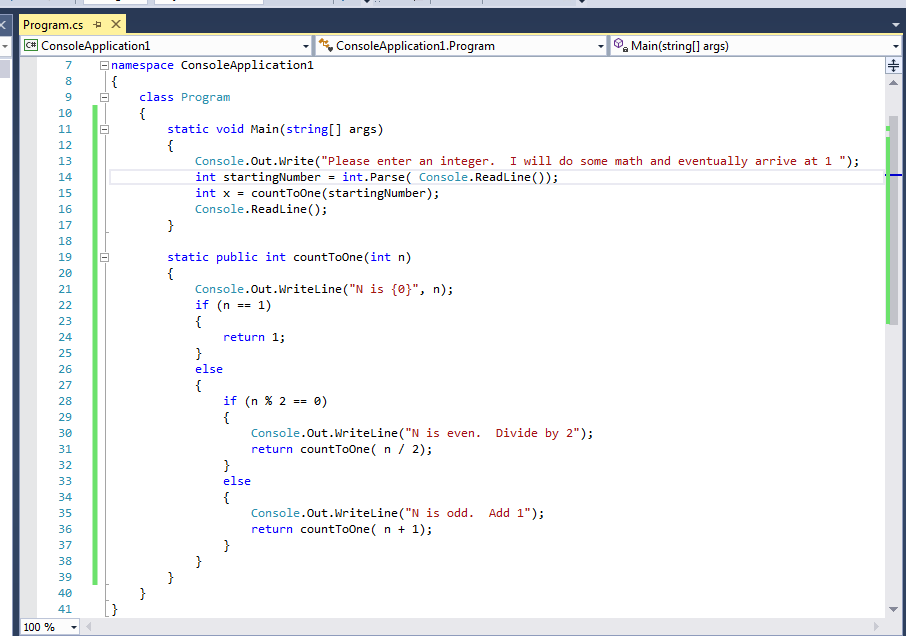
Recursion is an iterative way about and solving a problem where a function calls itself. Read more about recursion programming at <https://www.geeksforgeeks.org/recursion/> and <https://www.youtube.com/watch?v=neuDuf_i8Sg>.

# 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.



**Deliverables:**

1. Take a screenshot of the application being run and put it into a 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 x 4 ) x 3 x 2 x 1 = ( 20 x 3 ) x 2 x 1 = ( 60 x 2 ) x 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.

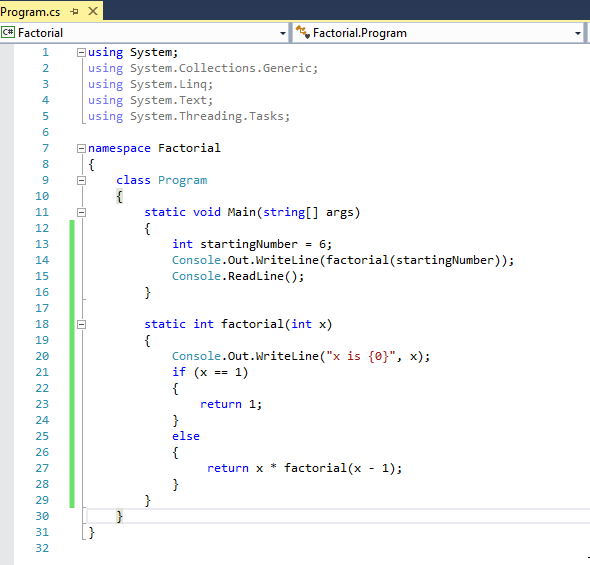
4! = 1\*2\*3\*4 or 4\*3\*2\*1 = 24

Secondly, we can use recursive thinking.

4! = 4 \* 3!  
= 4 \* (3\*2!)  
= 4 \* (3\*(2\*1!))  
= 4 \* (3\*(2\*(1)))  
= 24

ecursionmadesimple/Factorial_Recursive_Case.jpg

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.

:

**Deliverables:**

1. Take a screenshot of the application being run and put it into a 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 the divisors of 15 are 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

A **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 = 5

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:

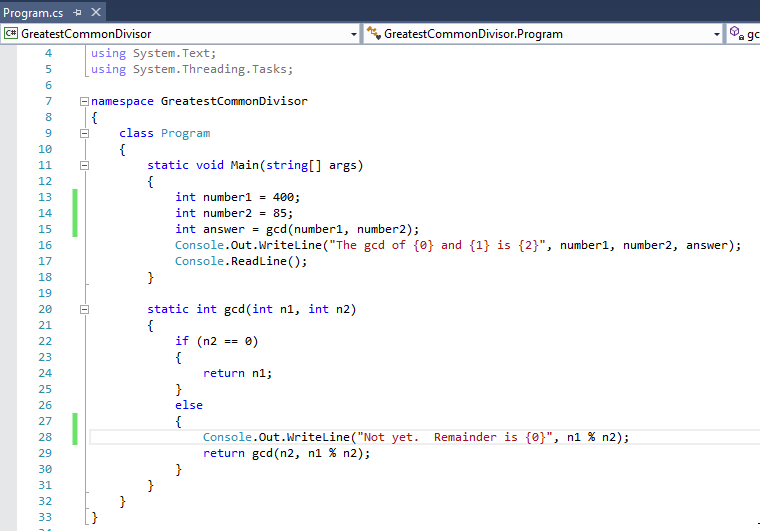
Find the GCD for 180 and 150.

|  |  |  |
| --- | --- | --- |
| 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:



**Deliverables:**

1. Take a screenshot of the application being run and put it into a 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.

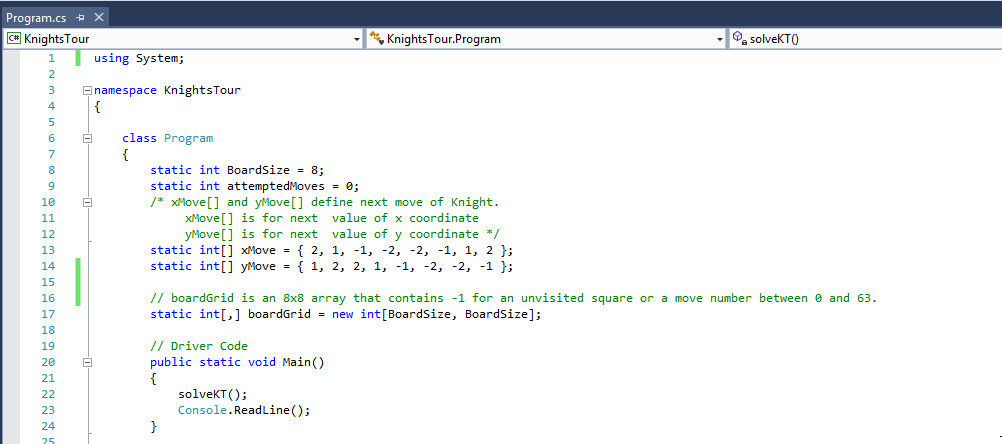
Look at the Wikipedia article for an animated example of how the computer solves the problem.

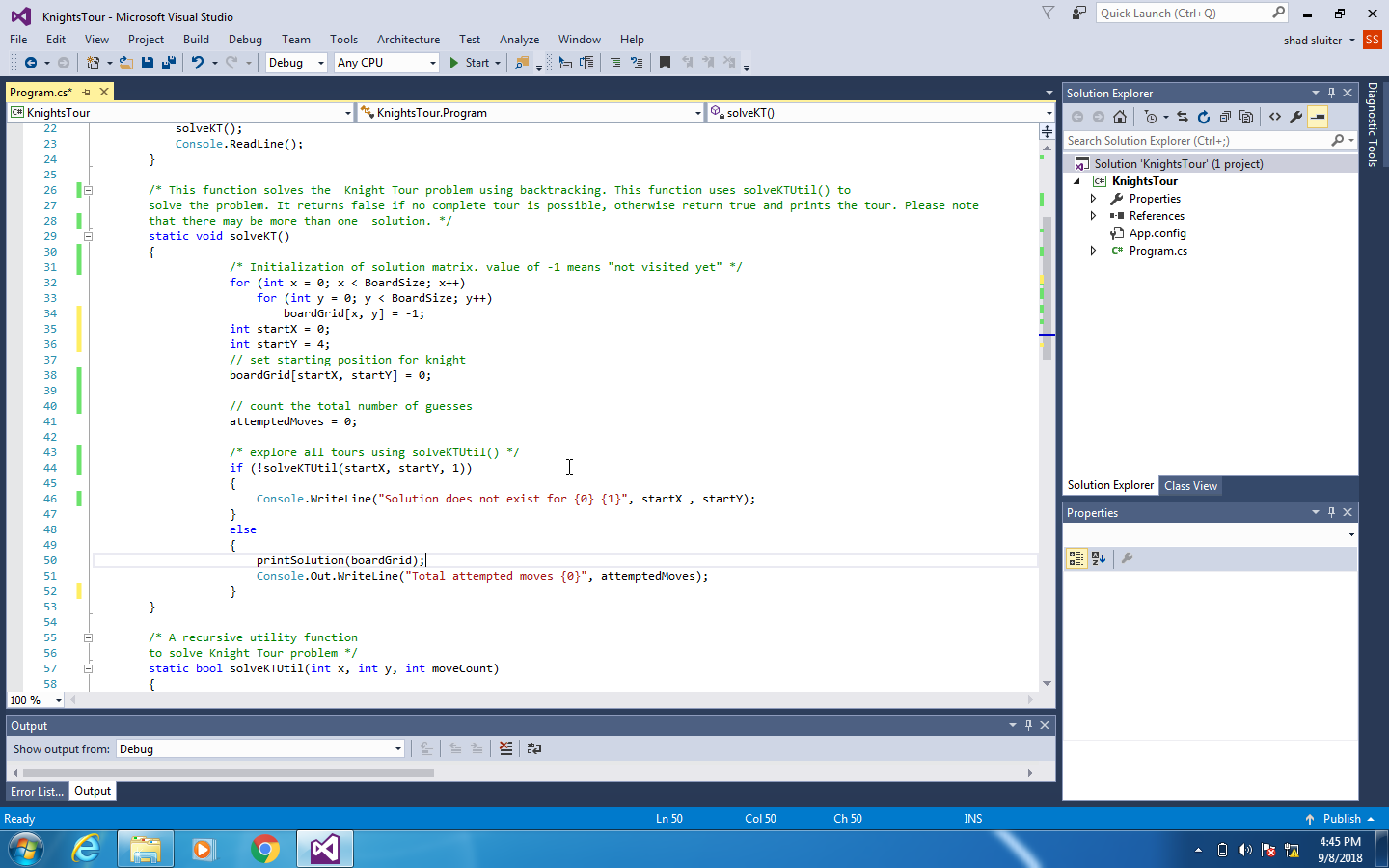
<https://en.wikipedia.org/wiki/Knight%27s_tour>

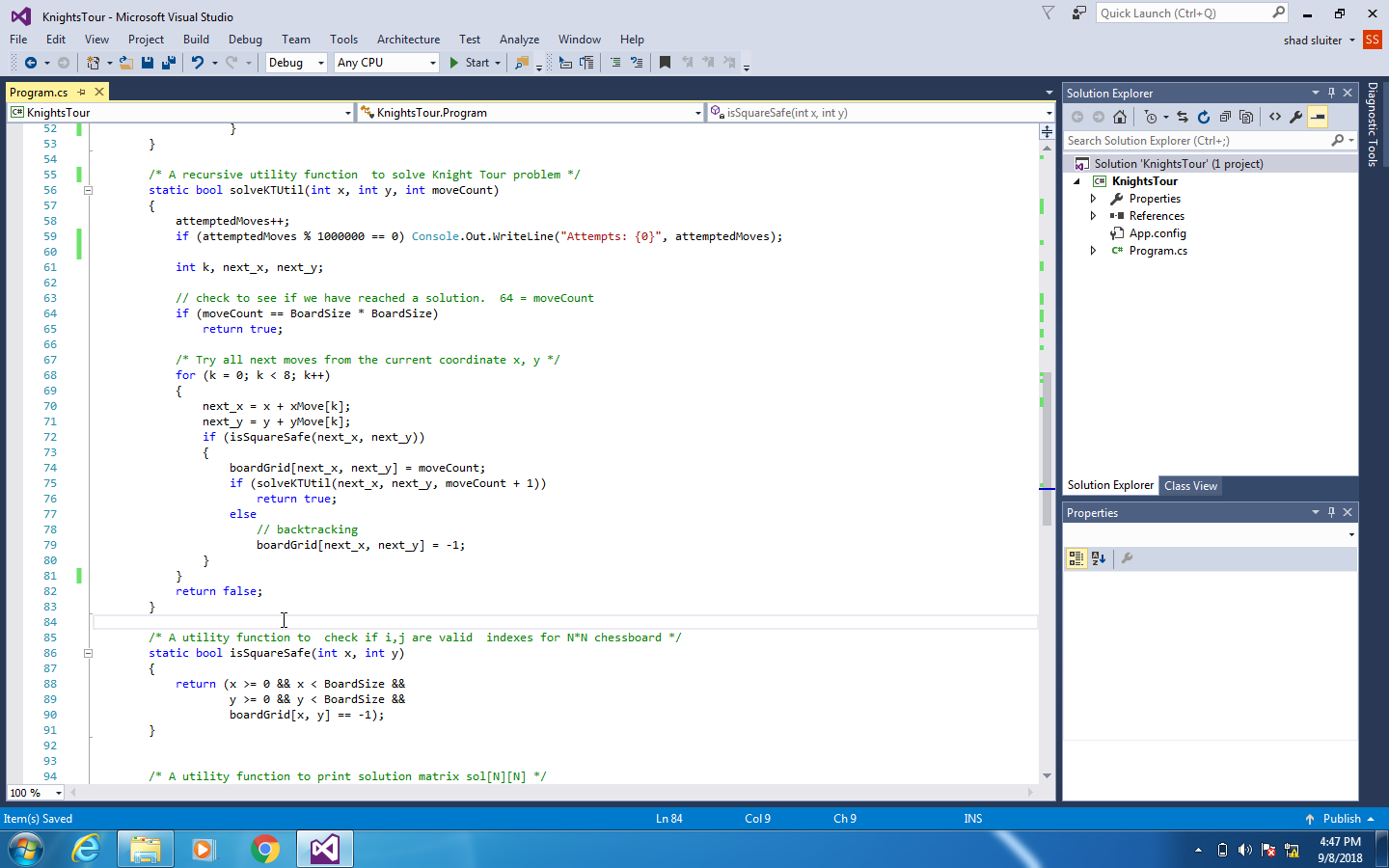
The solution we will look at is a "brute force" solver. That is, the compute will attempt every combination of moves until he finds a solution or concludes that no solutions exist.

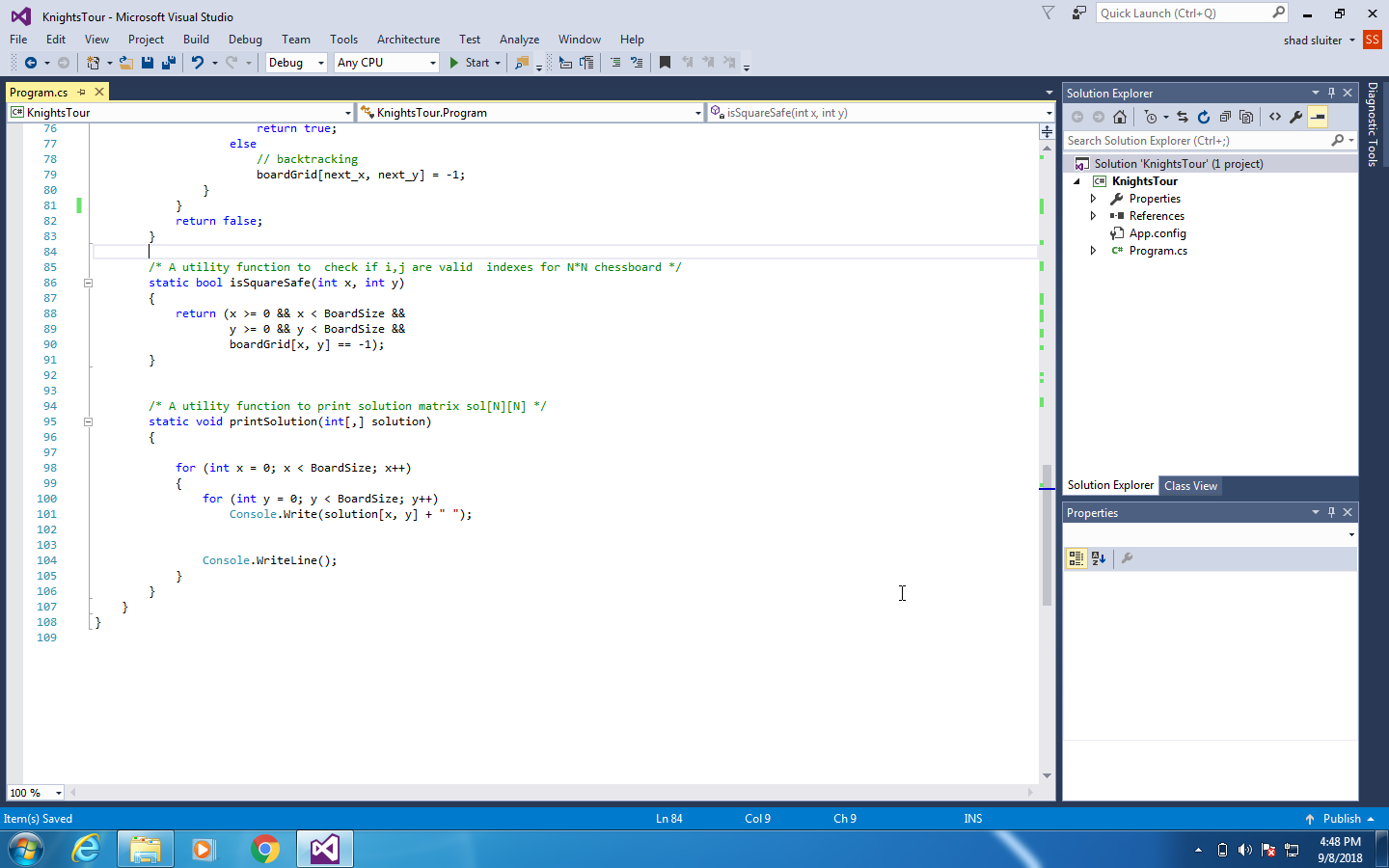
Here is the algorithm:

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.









**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.

Read about Warnsdorf's rule at "Knight's Tour": <https://en.wikipedia.org/wiki/Knight%27s_tour>.

See a demonstration of "Knight's Tour Warnsdorf Rule" at: <https://www.youtube.com/watch?v=9Gj91wJ8STU>.

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 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.