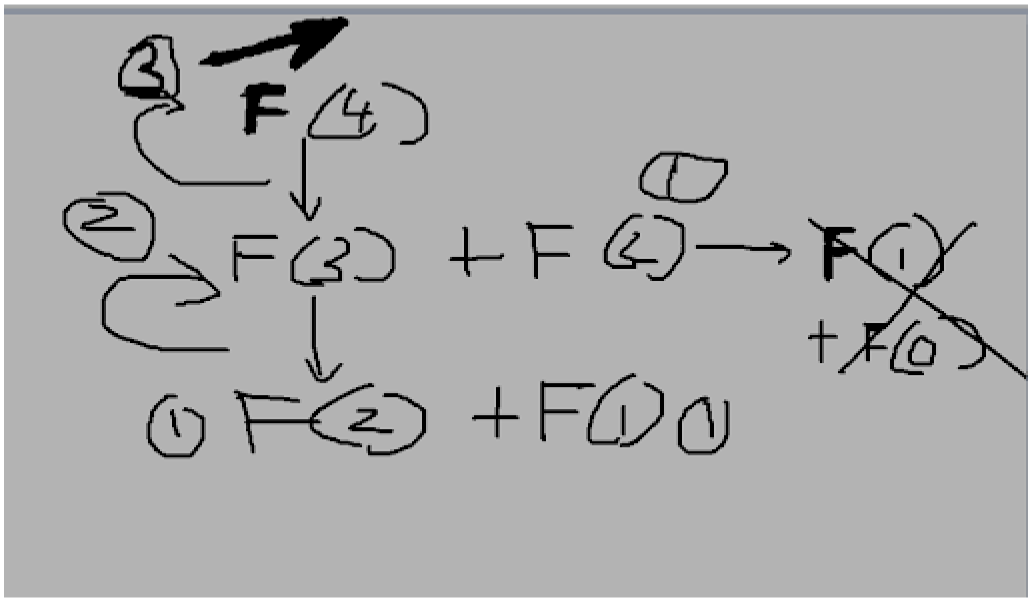
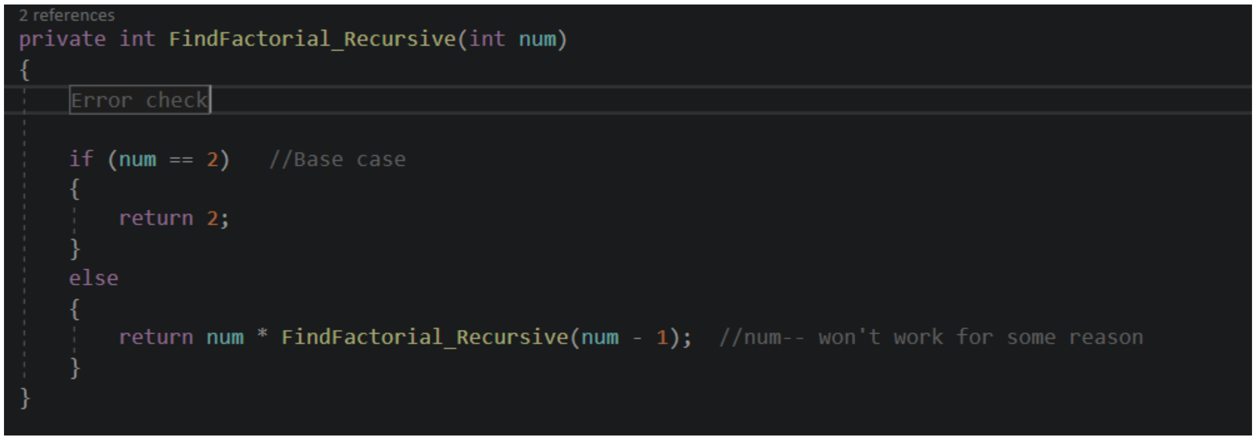
Recursive code and the call stack

Fibonacci

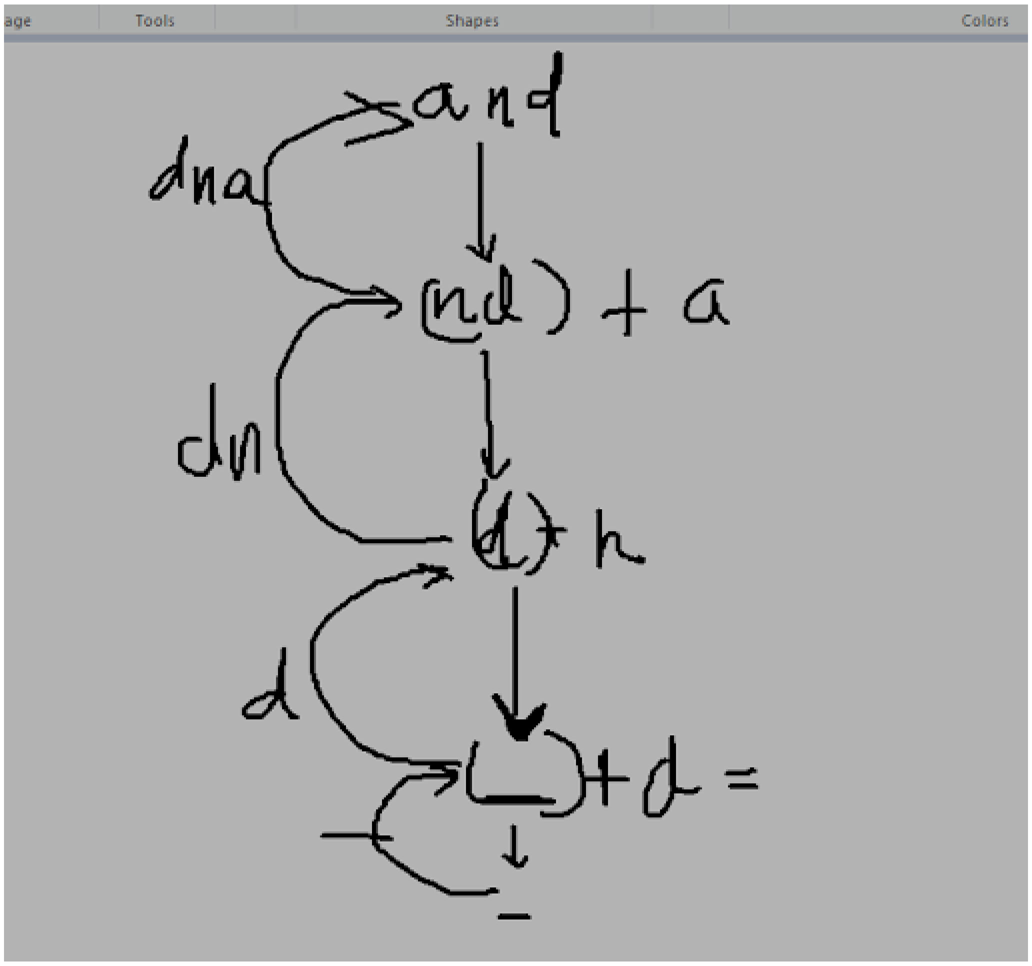


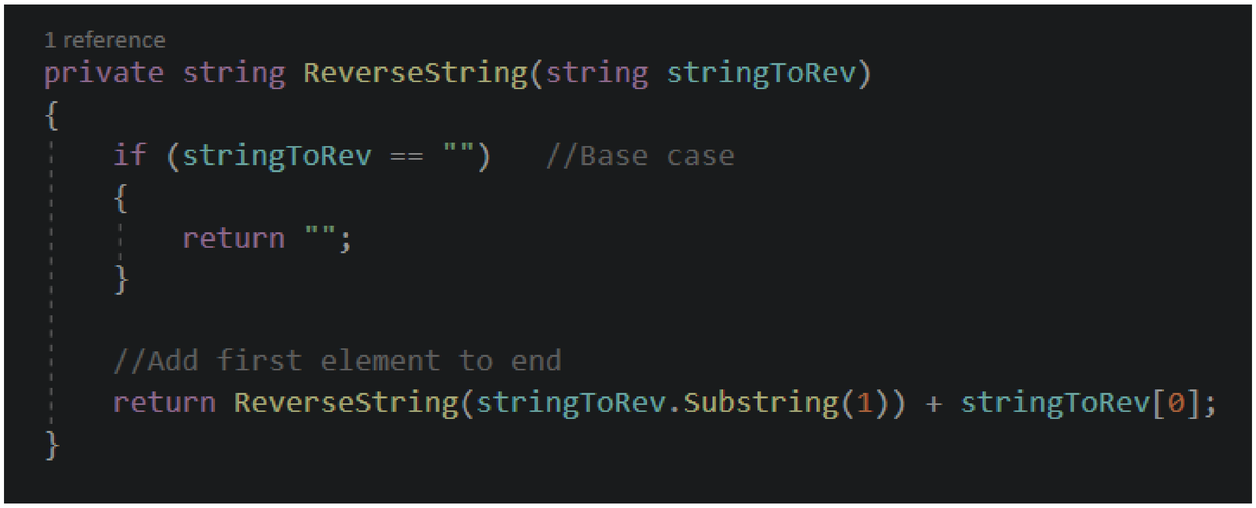


```
2 references
private int FindFactorial_Recursive(int num)
{
    Error check

    if (num == 2)   //Base case
    {
        return 2;
    }
    else
    {
        return num * FindFactorial_Recursive(num - 1);  //num-- won't work for some reason
    }
}
```

String Reversal

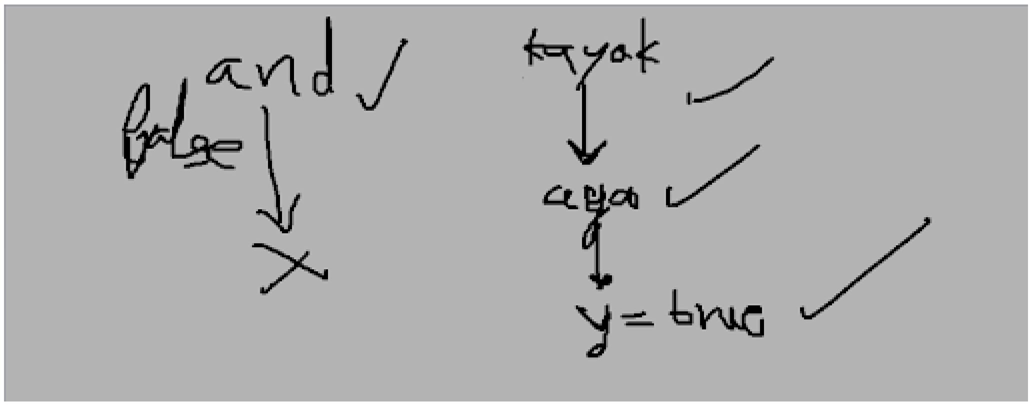


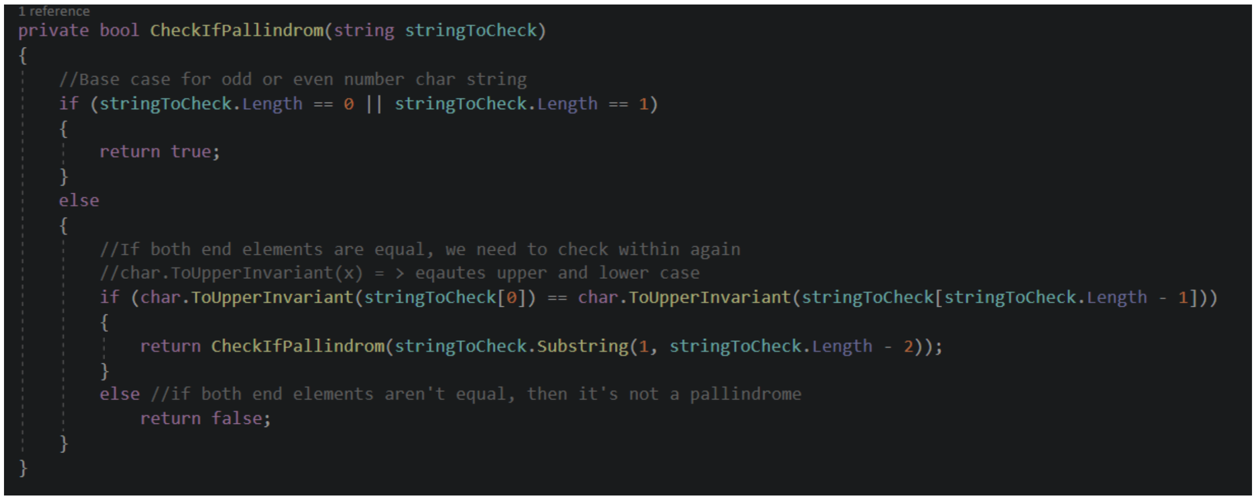


```
1 reference
private string ReverseString(string stringToRev)
{
    if (stringToRev == "")   //Base case
    {
        return "";
    }

    //Add first element to end
    return ReverseString(stringToRev.Substring(1)) + stringToRev[0];
}
```

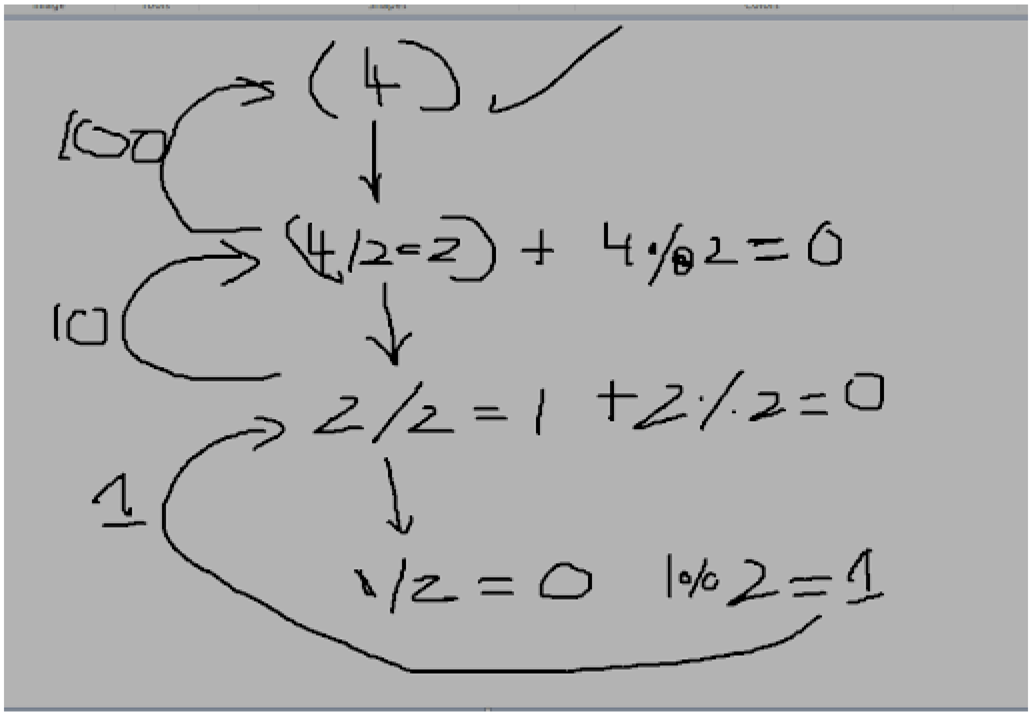
Palindrome check

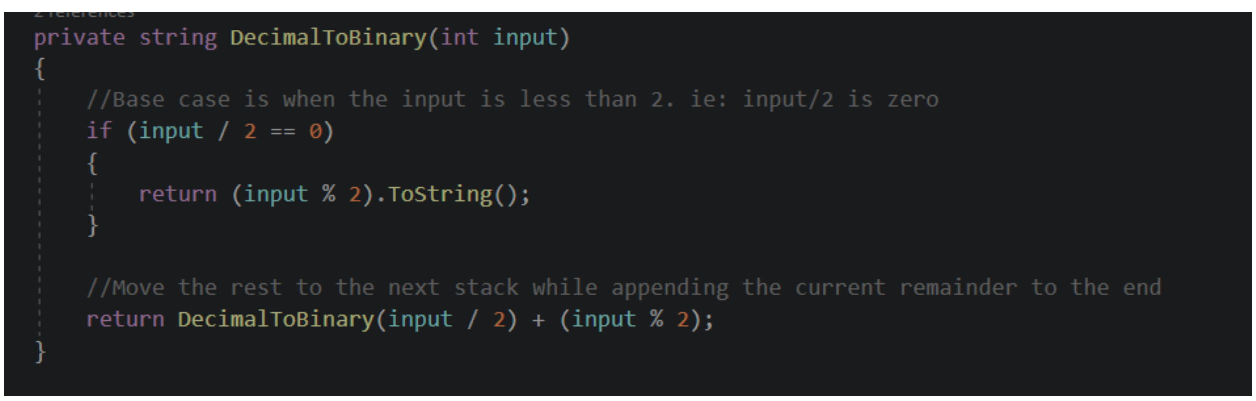




```
1 reference
private bool CheckIfPallindrom(string stringToCheck)
{
    //Base case for odd or even number char string
    if (stringToCheck.Length == 0 || stringToCheck.Length == 1)
    {
        return true;
    }
    else
    {
        //If both end elements are equal, we need to check within again
        //char.ToUpperInvariant(x) = > eqautes upper and lower case
        if (char.ToUpperInvariant(stringToCheck[0]) == char.ToUpperInvariant(stringToCheck[stringToCheck.Length - 1]))
        {
            return CheckIfPallindrom(stringToCheck.Substring(1, stringToCheck.Length - 2));
        }
        else //if both end elements aren't equal, then it's not a pallindrome
            return false;
    }
}
```

Binary to Decimal

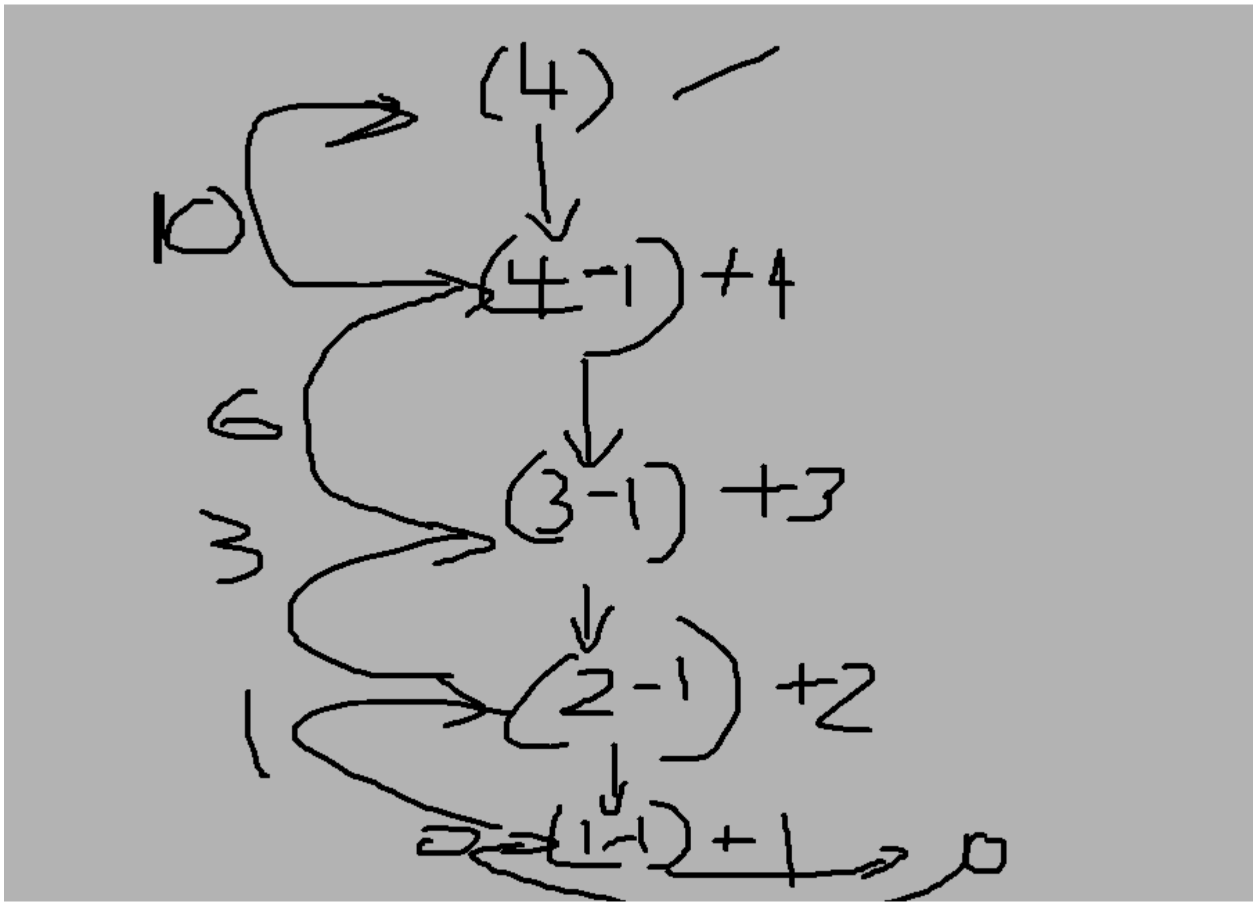


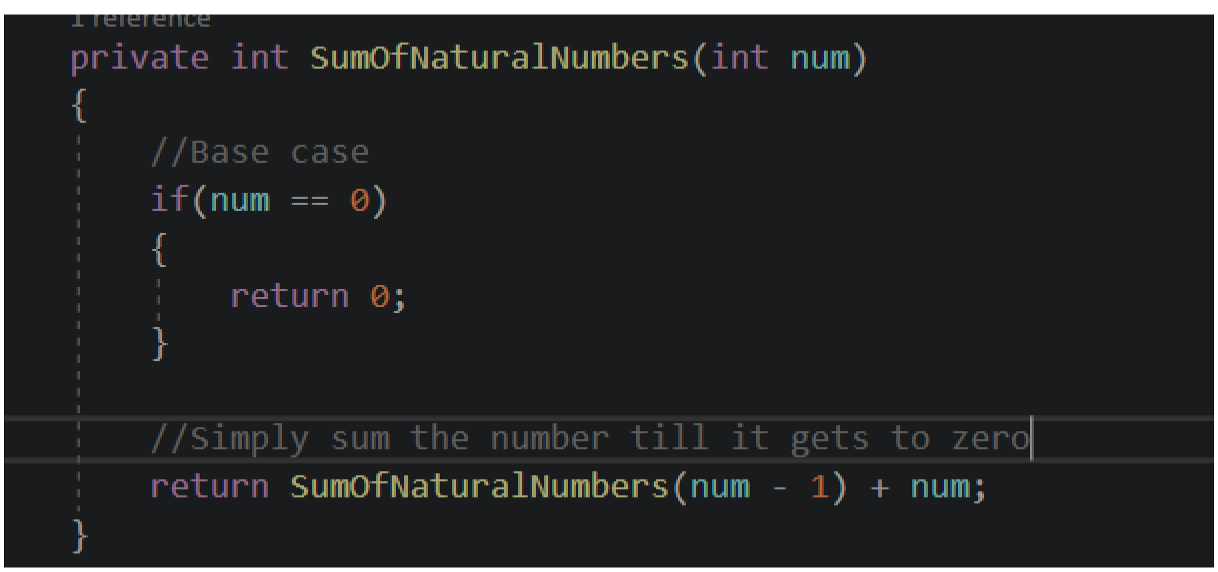


```
private string DecimalToBinary(int input)
{
    //Base case is when the input is less than 2. ie: input/2 is zero
    if (input / 2 == 0)
    {
        return (input % 2).ToString();
    }

    //Move the rest to the next stack while appending the current remainder to the end
    return DecimalToBinary(input / 2) + (input % 2);
}
```

Sum of n natural numbers





```
private int SumOfNaturalNumbers(int num)
{
    //Base case
    if(num == 0)
    {
        return 0;
    }

    //Simply sum the number till it gets to zero
    return SumOfNaturalNumbers(num - 1) + num;
}
```