CSC180 Assignment #04

Exercise 5.5. The point of this exercise is to use a stack diagram to understand the

execution of a recursive program.

public class Prod {

public static void Main(string[] args)

{Console.WriteLine(Prod(1, 4));}

public static int Prod(int m, int n) {

if ( m == n ) {

return n;

} else {

int recurse = Prod(m, n-1);

int result = n \* recurse;

return result;}}}

1. Draw a stack diagram showing the state of the program just before the last instance

of Prod completes. What is the output of this program?

Main



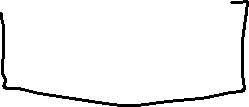
Prod m1, n4 recurse n6 result 24



Prod m1, n3 recurse n2 result 6



Prod m1, n2 recurse n1 result 2



Prod m1, n1



2. Explain in a few words what Prod does.

Makes a recursive call to find the factorial of n-1 and multiply it by n.

3. Rewrite Prod without using the temporary variables recurse and result.

using System;

namespace CSC180

{

public class prod

{

public static void Main(string[] args)

{

Console.WriteLine(Prod(1, 4));

}

public static int Prod(int m, int n)

{

if (m == n)

{

return n;

}

else

{

return n\*(Prod(m, n-1));

}

}

}

}

Exercise 6.2. Consider the following code:

static void Main(string[] args) {

double count = 0.0;

while ( count != 1.0 ) {

count = count + 0.1;

Console.Write("{0:R}", count);

Console.ReadLine();

}

}

Note: You will have to press the Enter or Return key for each iteration of the loop.

You will have to use Ctrl-C or close the window to stop the program.

1. Why does it not stop when count reaches 1.0?

The result of 1.0 was never reached in the factorial due to the doubles precision.

1. How can you alter it to do so?

using System;

namespace CSC180

{

public class Program

{

static void Main(string[] args)

{

decimal count = 0.0m;

while (count != 1.0m)

{

count = count + 0.1m;

Console.Write(count);

Console.ReadLine();

}

}

}

}

Exercise 5.7.

1. Create a program called Recurse.cs and type in the following methods:

// first: returns the first character of the given string

public static char First(string s) {

return s[0];

}

// rest: returns a new string that contains all but the

// first letter of the given string

public static string Rest(string s) {

return s.substring(1);

}

// length: returns the length of the given string

public static int Length(string s) {

return s.Length;

}

2. Write some code in Main that tests each of these methods. Make sure they work,

and make sure you understand what they do.

using System;

namespace CSC180

{

public class Program

{

public static char First(string s)

{

return s[0];

}

public static string Rest(string s)

{

return s.Substring(1);

}

public static int Length(string s)

{

return s.Length;

}

public static void Main()

{

string s = "abcdefg";

Console.WriteLine(First(s));

Console.WriteLine(Rest(s));

Console.WriteLine(Length(s));

}

}

}

3. Write a method called WriteString that takes a string as a parameter and that

prints the letters of the string, one on each line. It should be a void method.

using System;

public class Program

{

public static void Main(string[] args)

{

string s = "abcdefg";

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

{

Console.WriteLine(s[i]);

}

}

}

4. Write a method called WriteBackward that does the same thing as WriteString but

that prints the string backward (one character per line).

using System;

public class Program

{

public static void Main(string[] args)

{

string s = "abcdefg";

Console.WriteLine();

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

{

Console.WriteLine(s[s.Length - i - 1]);

}

}

}

5. Write a method called ReverseString that takes a string as a parameter and

that returns a new string as a return value. The new string should contain the

same letters as the parameter, but in reverse order.

using System;

public class Program

{

public static void Main(string[] args)

{

string s = "abcdefg";

string tmp = "";

for (int i = s.Length - 1; i >= 0; i--)

{

tmp = tmp + s[i];

}

Console.WriteLine(tmp);

}

}