Contents

Recursion (Solutions)

Multiple Choices

1. What is the simplest definition of recursion?

   ☐ A method is recursive if it does not take arguments.
   ☐ A property is recursive if it has a backing field.
   ☒ A method is recursive if it calls itself.
   ☐ A method is recursive if it calls a constructor.
   ☐ A class is recursive if it inherits from another class.

2. Consider the following code:

```
void Test(int n)
    {
    if (n != 0){
        Console.Write($"{n} ");
        Test(n - 1);
        }
    }
Test(3);
```

   What will it display?

   ☐ Nothing
   ☐ n
   ☒ 3 2 1
   ☐ 3 2 1 0
   ☐ 3 2 1 0 -1 -2 -3 -4 -5 … until the program crashes.
   ☐ 2 1 0 -1 -2 -3 -4 -5 … until the program crashes.
   ☐ 1 2 3
   ☐ 0 1 2 3

3. Consider the following code:

```
void Test(int n)
    {
        Console.Write($"{n} ");
        Test(n - 1);
    }
Test(3);
```

What will it display?

- ☐ Nothing
- ☐ n
- ☐ 3 2 1
- ☐ 3 2 1 0
- ☒ 3 2 1 0 -1 -2 -3 -4 -5 … until the program crashes.
- ☐ 2 1 0 -1 -2 -3 -4 -5 … until the program crashes.
- ☐ 1 2 3
- ☐ 0 1 2 3

4. Consider the following code:

```
void Test(int n)
    {
        Test(n - 1);
        Console.Write($"{n} ");
    }
Test(3);
```

What will it display?

- ☒ Nothing
- ☐ n
- ☐ 3 2 1
- ☐ 3 2 1 0
- ☐ 3 2 1 0 -1 -2 -3 -4 -5 … until the program crashes.
- ☐ 2 1 0 -1 -2 -3 -4 -5 … until the program crashes.
- ☐ 1 2 3
- ☐ 0 1 2 3

Exercises

1. What would the following code display?

```
int Myst1(int n)
{
    if (n != 0)
    {
        return n + Myst1(n - 1);
    }
    else
    {
        return n;
    }
}

Console.WriteLine(Myst1(4));
```

Solution

10

2. What would the following code display?

```
void Myst2(int n)
{
    if (n == 0) { Console.WriteLine("Done"); }
    else if (n < 0)
    {
        Console.Write($"{n} ");
        Myst2(-n);
    }
    else
    {
        Console.Write($"{n} ");
        Myst2(-(n - 1));
    }
}
Myst2(3);
```

Solution

3 -2 2 -1 1 Done

3. What would the following code display?

```
void Myst3(int len)
{
    MystH(0, 1, 1, len);
}
void MystH(int axP, int bxP, int counter, int len)
{
    if (counter <= len)
    {
        Console.Write($"{axP} ");
        MystH(bxP, axP + bxP, counter + 1, len);
    }
}
Myst3(6);
```

Solution

0 1 1 2 3 5

Those are the first 6 digits of the Fibonacci sequence[1].

4. Write a recursive method that takes an `int` as argument, generates a random `int` between `0` and this argument, displays it and

[1] https://en.wikipedia.org/wiki/Fibonacci_sequence

calls itself with that number. The method should stop when the `int` generated is 0.

Solution

```
void RandomTo0(int n)
{
    Random seed = new Random();
    int randomNumber = seed.Next(0, n);
    if (randomNumber == 0) {
     ↪  Console.WriteLine("Done"); }
    else
    {
        Console.WriteLine($"Number: {randomNumber}");
         RandomTo0(randomNumber);
    }
}
```

5. Write a recursive method that takes a `string` as argument and returns **`true`** if it is a palindrome. Your method should return **`true`** on input `"civic"`, `"noon"`, `"radar"` and `""` (empty string), and **`false`** on input `"test"` and `"not a palindrome"`.

   Hint

   Use `Length - 1` to access the last index of the string, and the `Substring` method to create a string without the first and last characters.

   Solution

```
bool Palindrome(string s)
{
    if (s.Length <= 1) return true;
    else
    {
        if (s[0] == s[s.Length - 1]) return
         ↪  Palindrome(s.Substring(1, s.Length -
         ↪  2));
        else return false;
    }
}

List<string> palindrome_test = new List<string> {
↪  "radar", "civic", "noon", "", "test", "not a
↪  palindrome"};
foreach (string test in palindrome_test)
{
```

```
    Console.WriteLine(test + " is a palindrome: " +
↪   Palindrome(test));
}
```

6. Write a recursive method that takes an `int` as argument and returns the number of even digits in it. For example, on input `631`, the method should return `1` since only `6` is even.

   Hint

   The `% 10` operation gives the last digit of a number (`631 % 10` gives `1`), and the `/ 10` operation (on integers) removes the last digit from an int (`631 / 10` gives `63`).

   Solution

```
int CountEven(int n)
{
    if (n == 0) { return 0; }
    else if (n % 10 % 2 == 0) { return 1 +
↪    CountEven(n / 10); }
    else { return CountEven(n / 10); }
}
```