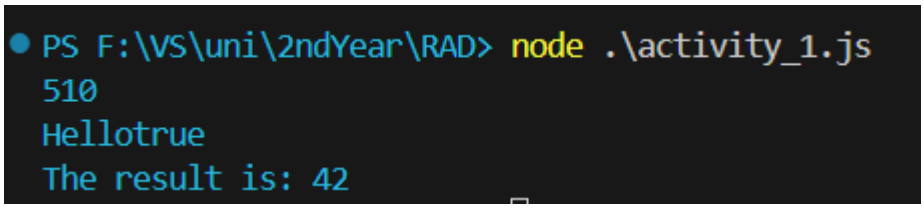
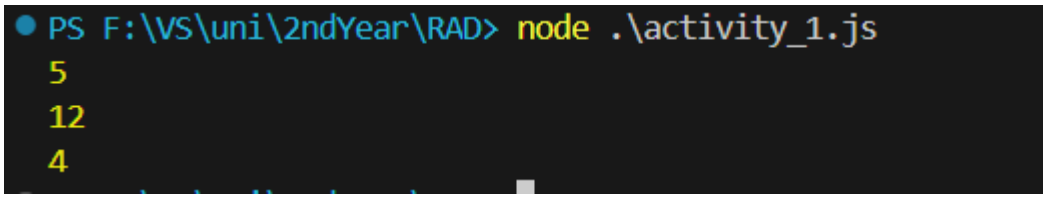
1. **String Coercion:**

In JavaScript, when the + operator is used with a string and another type (like a number or a Boolean), the non-string operand is automatically converted to a string because the + operator performs string concatenation when one of its operands is a string.



2. **Number Coercion:**

In arithmetic operations like subtraction (-), multiplication (*), and division (/), JavaScript converts string operands to numbers because these operators are defined for numerical operations. This type coercion allows JavaScript to perform the intended arithmetic calculations rather than concatenating strings.



3. **Boolean Coercion:**

When booleans are used in arithmetic operations, JavaScript converts true to 1 and false to 0 because it treats booleans as numeric values for the purpose of these calculations. This conversion allows JavaScript to seamlessly perform arithmetic operations involving boolean values.

```
PS F:\VS\uni\2ndYear\RAD> node .\activity_1.js
true
false
false
```

4. Null and Undefined Comparisons:

null and undefined are only equal to each other when using the loose equality operator (==) because they represent the absence of any object value and the absence of a value, respectively. They are not considered equal to any other values, such as 0, because their types and meanings differ fundamentally from other data types.

```
PS F:\VS\uni\2ndYear\RAD> node .\activity_1.js
3
0
1
```

5:

```
PS F:\VS\uni\2ndYear\RAD> node .\activity_1.js
2
true10
true
[object Object]test
1
```