Arithmetic with Strings

# 1. Algorithm adding two strings of numbers and presents the result as a new string.

## Problem

Two strings are provided, both containing natural numbers. These should be added together an the result stored in a new string, which is presented to the user.

## Algorithm

### Preconditions

Two natural numbers are given as inputs. The first is larger than the second and no negative numbers are allowed.

### Postconditions

The result of the addition is shown as a regular shorthand addition table

### Steps in the algorithm

Make helper variables, one for carry, one for intermediate results and one for total.

Take input 1 and input 2 and reverse both.

Take the first character in both inputs, save it as an int and subtract 48 (to convert to a singular integer)

Add these together with the carry.

If the resulting is larger than 9, set carry to 1 and take 10 out of the result.

Add result to the string total.

Repeat until each digit has been added.

If one of the numbers is longer than the other, keep appending the remaining digits to the total, otherwise do no more.

Reverse total and return it.

### Graphical representation of the algorithm:

First number

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | … | Xn | Xn+1 |

**+**

**+**

**+**

**+**

**+**

Second number

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Y1 | Y2 | Y3 | … | Yn | Yn+1 | Yn+2 |

**+**

**+**

**+**

**+**

**+**

**+**

Carry

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | carry1 | carry2 | … | carryn-1 | carryn | carryn+1 |

Result

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| res1 | res2 | res3 | … | resn | resn+1 | resn+2 |

## Pseudocode

carry = 0

result = 0

resultContainer = new StringBuilder()

total = “”;

input1.reverse()

input2.reverse()

longestNumber = ((input1.length() >= input2.length()) ? 1 : 2)

if longestNumber == 1

length = input2.length()

else

length = input1.length()

while i < length

x = input1.charAt(i) – 48

y = input2.charAt(i) - 48

result = x + y + carry

if result >= 10

result = result – 10

carry = 1

resultContainer.append(result)

switch (longestNumber)

case: 1

while i < input1.length()

x = input1[i] – 48

result = x + carry

resultContainer.append(result)

carry = 0

i++

case: 2

while i < input2.length()

x = input1.charAt(i) – 48

result = x + carry

resultContainer.append(result)

carry = 0

i++

total = resultContainer.reverse()

return total

# 2. Algorithm subtracting two strings of numbers and presents the result as a new string.

## Problem

Take two strings containing natural numbers. Subtract the second from the first. Return the result as a new string that is presented to the user.

## Algorithm

### Preconditions

Two natural numbers (n ≥ 0) are given as input by the user.

### Postconditions

The result of the subtraction can be found in the returned string.

### Steps in algorithm

Ask user for two numbers and save them as separate variables.

Compare the string lengths to see which one is larger.

Set the shorter length as the number of times to loop in a variable.

If both strings are equal, default to the first length as number of times to loop.

If both strings are equal in length, compare each character in turn to decide which number is larger. If both strings are the same, set result to 0 and return it. Once the larger number is found, save this as a variable called “larger” and the smaller number in a variable called “smaller”.

Reverse the strings to facilitate looping through them.

Compare the first digit in the largest (or first) number with the first digit in the second number.

If the first number is larger, take the first number and subtract the second and the carry. Set the carry to 0.

If the second number is larger, add 10 to the first number, subtract the second and a carry. Set the carry to 1.

Add the result to the result string.

If the carry is a 1 when loop finishes, and one number is larger than the other, take the next character in the larger number, subtract the carry, add the remaining digits to the result.

If the carry us 0 when loop finishes, add the remaining digits to the result.

If the second number was larger, add a “-“ to the result string.

Return the result.

### A graphical representation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10 or 0 | 10 or 0 | 10 or 0 | … | 10 or 0 | 0 | Borrowing |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **+** | **+** | **+** | **+** | **+** | **+** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | X3 | … | Xn | Xn+1 | The larger number |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **-** | **-** | **-** | **-** | **-** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Y1 | Y2 | Y3 | … | Yn |  | The smaller number |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **-** | **-** | **-** | **-** | **-** | **-** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | carry1 | carry2 | carry… | carryn | carryn+1 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **=** | **=** | **=** | **=** | **=** | **=** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Y1 | Y2 | Y3 | … | Yn |  | The result |

## Pseudocode

carry = 0

timestoloop = 0

negative = “”

result = “”

if num1.equals(num2)

result = “0”

return result

if num1.length = num2.length

i = 0

do

if num1.charAt(i) = num2.charAt(i)

i++

continue

else if num1.charAt(i) > num2.charAt(i)

larger = num1

smaller = num2

break

else

larger = num2

smaller = num1

negative = “-“

break

while i < num1.length

else if num1.length > num2.length

larger = num1

smaller = num2

else

larger = num2

smaller = num1

negative = “-“

timestoloop = smaller.length

larger = larger.reverse

smaller = smaller.reverse

i = 0 // reset i to 0 to use in new loop

while i < timestoloop

sub = 0

x = larger[i] – 48

y = smaller[i] – 48

if x < y

sub = 10 + x – y – carry

carry = 1

result.append(sub)

else

sub = x – y – carry

carry = 0

result.append(sub)

if carry = 1

x = larger.charAt[i] – carry

result.append(x)

i++

while i < larger.length

result.append(larger.charAt[i])

i++

result.append(negative)

return result.reverse.toString