Implementation Operations + Conversions

Problem Statement

The application implements algorithms for:

* Arithmetic operations for positive integers: addition, subtraction, multiplication and division by one digit, in a base p from {2,3,...,9,10,16}
* Conversions of natural numbers between two bases p, q from{2,3,...,9,10,16} using the substitution method or successive divisions and rapid conversions between two bases p, q from {2, 4, 8, 16}.

and lets the user verify them separately through a menu in the form of:

1. Addition;
2. Subtraction;
3. Multiplication by one digit;
4. Division by one digit;
5. Conversion:
   1. Substitution conversion b < h;
   2. Successive division conversion b > h;
   3. Rapid conversion;
   4. Intermediate base conversion;
   5. Quit.
6. Quit.

Note: The algorithms are tested using assert tests and the example data given in the course.

Contents

[Addition 3](#_Toc121689331)

[Subtraction 3](#_Toc121689332)

[Multiplication by one digit 4](#_Toc121689333)

[Division by one digit 4](#_Toc121689334)

[Conversion 5](#_Toc121689335)

[Substitution conversion b < h 5](#_Toc121689336)

[Successive division conversion b > h 5](#_Toc121689337)

[Rapid conversion 6](#_Toc121689338)

[Intermediate base conversion 6](#_Toc121689339)

# Addition

Problem statement

Add two numbers in a given base.

The used algorithm

1. We receive two numbers in a common base.
2. We iterate through them adding digit by digit from right to left and the carry (if there is any).

Implementation considerations

* In the case that the numbers have different lengths, we handle it by setting the digit to 0 (thus avoiding an “index out of range error”);
* The digits need to be converted into base 10 because the addition is done in base 10;
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings.
* For each iteration, the obtained result needs to be added to the left of the previous result.

# Subtraction

Problem statement

Subtract two numbers in a given base.

The used algorithm

1. We receive two numbers in a common base.
2. We iterate through them subtracting digit by digit from right to left and the carry (if there is any).

Implementation considerations

* In the case that the numbers have different lengths, we handle it by setting the digit to 0 (thus avoiding an “index out of range error”) ;
* The digits need to be converted into base 10 because the subtraction is done in base 10;
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings;
* The carry is handled either by subtracting it from the base or resetting it to 0.
* For each iteration, the obtained result needs to be added to the left of the previous result.

# Multiplication by one digit

Problem statement

Multiplication between a number and a digit in a given base.

The used algorithm

1. We receive two numbers in a common base.
2. We iterate through the first from right to left and multiply each digit by the second number. To this result we add the carry and calculate the carry for the next iteration

Implementation considerations

* In the case that the numbers have different lengths, we handle it by setting the digit to 0 (thus avoiding an “index out of range error”);
* The digits need to be converted into base 10 because the multiplication and addition are done in base 10;
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings;
* For each iteration, the obtained result needs to be added to the left of the previous result.

# Division by one digit

Problem statement

Division between a number and a digit in a given base.

The used algorithm

1. We receive two numbers in a common base.
2. We iterate through the digits of the dividend from left to right.
3. Divide the sum of the remainder from the previous iteration (if there is any) and the current digit by the divisor.
4. Obtain a new quotient and remainder. The quotient is added to the left of the previous quotient.

Implementation considerations

* The digits need to be converted into base 10 because the division, multiplication and addition are done in base 10;
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings;
* The result of the division contains two elements: the quotient and the remainder so a pair combining the two will be returned.

# Conversion

## Substitution conversion b < h

Problem statement

Conversion of a number from base b to base h, where b < h. Using substitution conversion.

The used algorithm

1. We receive the number, the source base and the destination base.
2. We iterate through the number from right to left taking one digit at a time.
3. Multiply this digit by the source base raised to the power of the digit’s position (calculation is done in the destination base)
4. Add up the results of all iterations.

Implementation considerations

* The multiplication and addition are done in the destination base;
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings;

## Successive division conversion b > h

Problem statement

Conversion of a number from base b to base h, where b > h. Using successive division conversion.

The used algorithm

1. We receive the number, the source base and the destination base.
2. We successively divide the number by the destination base, obtaining a quotient and a remainder.
3. The number repeatedly becomes the quotient until the quotient is 0.
4. The remainder is added to the left of the previous result.

Implementation considerations

* The division is done in the source base;
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings;
* For each iteration, the obtained result needs to be added to the left of the previous result.

## Rapid conversion

Problem statement

Conversion of a number from base b to base h, where both b and h are multiples of 2. Using rapid conversions.

The used algorithm

1. We need to decide how many digits to take from source base and to how many digits they correspond to in the destination base:
   1. If b < h, digits are taken and turned into 1, otherwise, 1 digit is taken and turned into digits

Implementation considerations

* Each digit/s from base b has/have a correspondent group of digits in base h, if both b and h are from {2, 4, 8, 16};
* When appending groups of digits, there needs to be a condition for filler 0’s to the left.
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings;
* For each iteration, the obtained result needs to be added to the left of the previous result.

## Intermediate base conversion

Problem statement

Conversion of a number from base b to base h. Using an intermediate base.

The used algorithm

1. We receive the number, the source base and the destination base.
2. Convert the number from the source base to base 10.
3. Convert the number from base 10 to the destination base.

Implementation considerations

* The conversion method used was substitution method.
* Since A-F cannot be interpreted as digits in python, the numbers will be handled as strings.