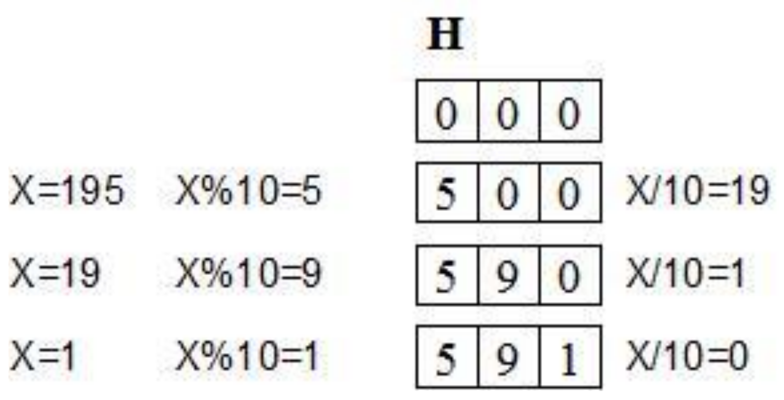
Student: Mircea Sorin-Sebastian
Group: 915

Documentation
For Operations and Conversions Application

The philosophy behind my implementation

I have used the principles of *big numbers* from the algorithmic domain. In this way every number it is split into digits and every digit occupies one position in a vector in a reverse way (in this manner the extra digits resulted from addition and multiplication are always added at the back of the vector resulting in a better complexity: O(n)).

For example, the number 195 in base 10 is represented in this way:



The main logic of the project is written under a class named numerationBases which provides all the necessary functionality.

Available methods

Data structure methods

- *base16NumberToVector(number)* - transforms a number with the base greater than 10 into a vector respecting the above principle.
- *numberToVector(number)* - transforms a number with the base less or equal to 10 into a vector respecting the above principle.
- *vectorToBase16Number(vector)* - transforms a vector intro a string representing the number (used for bases greater than 10 in order to express the digits ’10’ as ‘A’, ’11’ as ‘B’ and so on.
- *vectorToNumber(vector)* - transforms a vector into an number when base is equal or less than 10.

Operation methods

- *sum(a, b, numerationBase)* where *a* and *b* are already vectors - returns the result of addition between *a* and *b.*
- *sub(a, b, numerationBase)* where *a* and *b* are already vectors - returns the result of subtraction between *a* and *b.*
- *prod(a, nr, numerationBase)* where *a* is a vector and *b* is a digit - returns the result of the multiplication between *a* and *nr.*
- *div(a, nr, numerationBase)* where *a* is a vector and *b* is a digit - returns the quotent of the division between *a* and *nr.*
- *divR(a, nr, numerationBase)* where *a* is a vector and *b* is a digit - returns the remainder of the division between *a* and *nr.*

Conversion methods (rational numbers are accepted)

- convertFromBase10*(nr, baseDst)* - converts *nr* from base 10 into *baseDst* (destination base) using the successive divisions and multiplications method
- convertToBase10(nr, baseSrc) - converts *nr* from *baseSrc* (source base) into base 10 using the substitution method
- rapidConversionToLowerBase(a, baseSrc, baseDst) - converts *nr* from *baseSrc* (source base) into *baseDst* (destination base) using the method of rapid conversion.
    - if *baseSrc* is 2 than *baseDst* is either 4 or 8 or 16
    - if baseSrc is 4 than *baseDst* must be 16
- rapidConversionToHigherBase(a, baseSrc, baseDst) - converts *nr* from *baseSrc* (source base) into *baseDst* (destination base) using the method of rapid conversion.
    - if *baseSrc* is 16 than *baseDst* is either 8 or 4 or 2
    - if baseSrc is 4 than *baseDst* must be 2
- conversionToHigherBase(a, baseSrc, baseDst) - converts *nr* from *baseSrc* (source base) into *baseDst* (destination base) directly (without going through base 10) using the substitution method. Also baseSrc is smaller than baseDst
- conversionToLowerBase(a, baseSrc, baseDst) - converts *nr* from *baseSrc* (source base) into *baseDst* (destination base) directly (without going through base 10) using the successive divisions and multiplications method. Also baseSrc is greater than baseDst.

Operations pseudocode

Addition

```
FUNCTION sum(self, a, b, base):
    '''
    :param a: vectOR
    :param b: vectOR
    :param base: numeration base
    :return: a vector containing the sum of a AND b
    '''
    result <- new list
    t <- 0
    for i in range(length(a) - length(b)):
        b.append(0)
    ENDFOR
    for i in range(length(b) - length(a)):
        a.append(0)
    ENDFOR
    for i in range(max(length(a), lengt(b))):
        result.append((a[i] + b[i] + t)%base)
        t <- (a[i] + b[i] + t) // base
    ENDFOR
    IF t:
        result.append(t)
    ENDIF
    RETURN result
ENDFUNCTION
```

Subtraction

```
FUNCTION sub(self, a, b, base):
    '''
    :param a: vector (is greater than b)
    :param b: vectOR
    :param base: numeration base
    :return: a vector containing the substraction of a AND b
    '''
    result <- new list
    t <- 0
    for i in range(length(a) - length(b)):
        b.append(0)
    ENDFOR
    for i in range(max(length(a), length(b))):
        newRes <- a[i] - b[i] - t
        IF newRes < 0:
            t <- 1
            newRes <- newRes + base
        ELSE:
            t <- 0
        ENDIF
        result.append(newRes)
    ENDFOR
    while result[len(result) -1] = 0:
        result.pop(len(result) - 1)
    ENDWHILE
    RETURN result
ENDFUNCTION
```

Multiplication

```
FUNCTION prod(self, a, nr, base):
    '''
    :param a: vectOR
    :param nr: a number with 1 digit in specified base
                                                ENDIF
    :param base: numeration base
    :return: a vector containing the multiplication of a with base
    '''
    result <- new list
    t <- 0
    for i in range(length(a)):
        newRes <- a[i] * nr + t
        result.append(newRes % base)
        t <- newRes // base
    ENDFOR
    while t:
        result.append(t % base)
        t <- t // base
    ENDWHILE
    RETURN result
ENDFUNCTION
```

Division

```
FUNCTION div(self, a, nr, base):
    '''
    :param a: vectOR
    :param nr: a number with 1 digit in specified base
                                                ENDIF
    :param base: numeration base
    :return: a vector containing the result of a / nr
    '''
    result <- new list
    t <- 0
    for i in range(length(a)-1, -1, -1):
        t <- t * base + a[i]
        newRes <- t // nr
        result.append(newRes)
        t <- t % nr
    ENDFOR
    while length(result) AND result[0] = 0:
        result.pop(0)
    ENDWHILE
    RETURN result[::-1]
ENDFUNCTION
```

Test cases

I. 1404(5)+310(5) = 2214 (5)
II. 1012(8)*4(8) = 4050 (8)
III. 1982(10) = 7BA (16)
IV. 10111101.11101010(10) = BD.EA (8)
V. 97A3.15(11) = 403224.0312 (5)