Operations & Conversions App

The application must implement algorithms for:

* arithmetic operations in a base {2, 3, ..., 9, 10, 16} for positive integers such as:
  + 1.1. Addition
  + 1.2. Subtraction
  + 1.3. Multiplication by one digit
  + 1.4. Division by one digit
* conversions of natural numbers between two bases from {2, 3, ..., 9, 10, 16} using:
  + 2.1. Substitution method
  + 2.2. Successive divisions and multiplications
  + 2.3. Rapid conversions between {2, 4, 8, 16}

Implementation:

* 1. O imagine care conține text, captură de ecran, software, Font

     Descriere generată automatO imagine care conține text, captură de ecran, software, Software multimedia

     Descriere generată automatAddition

This method computes the sum of two numbers in a given base. It takes three parameters: **base**, **number1**, and **number2**. The **base** parameter specifies the base of the number system being used, while **number1** and **number2** are the numbers to be added. The method first converts **number1** and **number2** into their internal representations using the *get\_representation()* method. Then, it initializes the **list\_answer** as a list of '0's with the same length as the internal representations. The converter dictionary is used to map characters to their corresponding numerical values in the given base. The method then iterates over the digits of **list\_first** and **list\_second** in reverse order. For each digit, it adds the corresponding digits from **list\_first**, **list\_second**, **and list\_answer**, the sum is computed using the converter dictionary, and the result is stored in **list\_answer[i]**. If i > 0 (not the leftmost digit), the carry is computed and stored in **list\_answer[i-1]**. Finally, the method joins the digits in **list\_answer** and strips leading zeros to obtain the final sum.

O imagine care conține text, captură de ecran, Font

Descriere generată automat1.2. Subtraction

O imagine care conține text, captură de ecran

Descriere generată automat*get\_representation(x, y)*: This function takes two string input parameters (x and y) and returns a tuple. It first determines the maximum length of x and y and pads with zeros (**zfill**) to the left of the shorter input strings to match the determined length. The resulting strings are all of the same length, which helps in later computation.

This method performs subtraction between two numbers in the given base number system. It takes the base of the number system, the numbers to subtract, and returns the result of the subtraction as a string. The method begins by obtaining the representations of **number1** and **number2** in the given base number system using the helper function *get\_representation ()*. It then initializes two lists: **list\_borrows** to keep track of borrow values during the subtraction process, and **list\_answer** to store the resulting digits of the subtraction. A dictionary called **converter** is created to map the digits and letters used in the number system to their corresponding values. The keys of the dictionary are the digits and letters, and the values are the corresponding numeric values. Another dictionary called **converter\_inv** is created to map the numeric values back to their corresponding digits and letters. The subtraction process is performed digit by digit from right to left. For each digit, the method computes the subtraction by subtracting the corresponding digits of **number2** from **number1**, along with any borrows from the previous digits. If the result is non-negative, the corresponding digit in **list\_answer** is set to the converted value. Otherwise, a borrow is obtained by subtracting the base value from the result, and the corresponding digit in **list\_answer** is set to the converted value. Finally, the **list\_answer** is converted into a string, any leading zeroes are stripped, and the resulting string is returned as the final result of the subtraction.

O imagine care conține captură de ecran, text

Descriere generată automat1.3. Multiplication by one digit

This function *get\_representation(x)* is used to convert a string into a list of characters with a lead '0'. The string x is assumed to represent a number in a specific base (ranging from binary up to hexadecimal).

O imagine care conține text, captură de ecran, software

Descriere generată automat

The function *compute\_multiplication ()* is taking in three parameters: **base**, **number**, **multiplier**. The method uses the *get\_representation ()* function to convert **number** into a list of digits. It then initializes two lists: **list\_carries** and **list\_answer**, both containing '0' characters. The method also initializes a **converter** dictionary that maps characters to their decimal values in the specified base. The dictionary is then used to compute the value of each digit in the multiplication process. The method iterates over the digits of **list\_x** (in reverse order) and computes the product of the corresponding digits in **list\_x** and **multiplier**, taking into account any carry-over from previous digits. The computed value is then converted back to the specified base using the **converter\_inv** dictionary. The method updates the **list\_answer** and **list\_carries** lists accordingly and returns the final result as a string representation of a number in the specified base. Leading zeroes are stripped from the result using the ***lstrip()*** method.

1.4. Division by one digit

O imagine care conține text, Font, captură de ecran

Descriere generată automat

This function *get\_representation(x)* is used to convert a string into a list of characters. The string x is assumed to represent a number in a specific base (ranging from binary up to hexadecimal).

O imagine care conține text, captură de ecran

Descriere generată automat

This method performs division in a given base number system. It takes in three parameters - **base**, **number**, and **divisor**. The method first converts the given number to a list representation using the *get\_representation ()* function. It initializes the **carry** as '0' and creates an empty list to store the division result. It then creates a **converter** dictionary to map characters from the number system to their respective values. The **converter\_inv** dictionary is created to map values to their respective characters in the number system. The method iterates through each digit in the list representation of the number. For each digit, it calculates the value by multiplying the carry by the base and adding the value of the current digit. It appends the character representation of the division result to the **list\_answer** list. It also updates the carry with the remainder of the division. After the iteration, the method joins the characters in the **list\_answer** list to form a string. It removes leading zeros from the string using the ***lstrip ()*** function. Finally, the method returns a tuple containing the division result as a string and the remaining carry.

O imagine care conține text, Font, captură de ecran

Descriere generată automat2.1.Substitution method

This function converts a given string 'x' into a list of its individual characters. It also adds a '0' at the beginning of the list to facilitate subsequent arithmetic operations.

O imagine care conține text, software, Software multimedia, captură de ecran

Descriere generată automat

This function converts a single character into its integer value. If the character is a digit, it is converted directly into its integer value. If the character is a letter, it is converted into an integer using ASCII values, with 'A' representing 10, 'B' representing 11, and so forth.

O imagine care conține text, captură de ecran, software

Descriere generată automat

This function converts an integer into its character representation. If the integer is less than 10, it is converted directly into a stringified digit. If it is 10 or more, it is converted into a character, with 10 representing 'A', 11 representing 'B', and so forth.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat**Check 1.3 for the explanation**

**Check 1.1 for the explanation**

O imagine care conține text, captură de ecran, software

Descriere generată automat

The ***substitution()*** function converts a number from one base to another. It takes three parameters: **x** (the number to be converted), **base1** (the base of the number to be converted), and **base2** (the base to which the number will be converted). If **base1** is not equal to 10, the function converts the number using a general conversion algorithm. The algorithm involves converting the number to base 10, and then converting it from base 10 to **`base2`.** The function uses helper functions `***get\_representation\_for\_operations ()*`,** `**m*ultiplication****()`,* and `***sum()***` to perform the conversions. The result is returned as a string. If `**base1`** is equal to 10, the function converts the number to a hexadecimal representation. The function iteratively divides the number by 16 and appends the remainder to the answer, until the number becomes 0. The resulting hexadecimal representation is returned as a string, with leading zeros removed. If the original number or the converted number is 0, the function returns the string "0".

O imagine care conține text, software, Software multimedia, captură de ecran

Descriere generată automat2.2.Successive divisions and multiplications

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat The ***division()*** function performs a division operation of a number (which is in a specific base) by another divizor. The dividend (number) is passed as a string and converted to a list called **list\_number**. Division is done digit-wise and the individual digits of the quotient are stored in the **list\_answer** array and the remainder in the **carrie** variable. Finally a tuple of strings is returned representing the result of the division (quotient and remainder).

If the **base\_to\_convert** is not 10, the algorithm converts the number **`x`** from **`base1`** to **`base\_to\_convert`.** It starts by performing successive divisions on **`x`** using **`base1`** as the divisor and **`base\_to\_convert`** as the base. The remainders of the divisions are stored and concatenated to form the converted number. If the **`base\_to\_convert`** is 10, the algorithm converts the number **`x`** from base 16 (hexadecimal) to base 10 (decimal). It iterates over each digit of **`x`,** multiplies the current result by 16, and adds the decimal value of the digit.

2.3.Rapid conversions

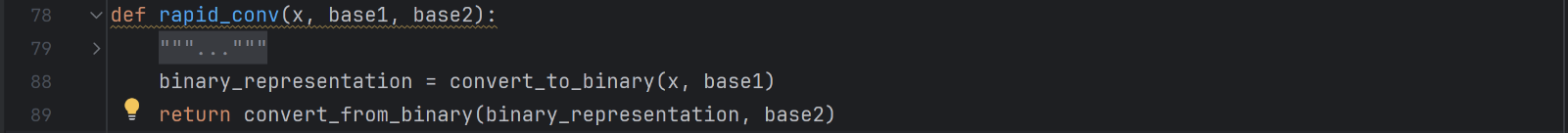
O imagine care conține text, captură de ecran, software

Descriere generată automatO imagine care conține text, Software multimedia, captură de ecran

Descriere generată automat This function calculates the highest power of 2.It does this by continuously dividing base by 2 until it is no longer divisible. The function returns the highest power of 2 which divides base.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automatThe function begins by breaking the string number into individual digits and initializing a **list**, **binary\_value**, to hold the resulting binary number. For every digit in the string number, the function uses a dictionary to map the digit to its decimal equivalent. It then uses a manner of successive division by two to convert each decimal number equivalent to binary. In each iteration, the binary equivalent of each digit is calculated and at the end of each calculation, added to the **binary\_value** list. Once the string number has been iterated through and all digits have been represented in binary, the binary list is joined into a string and any leading zeros from the binary representation are stripped and the resulting binary string is returned.

 The function works by iterating through each character in x, and for each character, extracting its value from a defined dictionary and transforming it into a binary representation. The function uses a dictionary to map base digits to their decimal values. For the base digit character, it executes a loop where it continuously divides the decimal value by 2 until it becomes 0 - creating a binary equivalent in the process. The function has a provision to handle numbers with bases greater than 10 - like hexadecimal numbers.