# Лабораторна робота №8

# Піддубняк Д. 6.1219-2

# Сложение(вычитание) чисел в дополнительном коде  
# Поддубняк Д. 6.1219-2  
# Функции: \_\_init\_\_ = конструктор класса,обьявляет переменные, binarconverter = конвертор в двоичный вид,  
# straight\_code = перевод в прямой код, reverse\_code = перевод в обратный код, additionally\_code = перевод в  
# дополнительный код, solution, substraction = сложение,вычитание  
#  
# Объявляем класс "Числа"  
class Digits:  
 def \_\_init\_\_(self, first\_num, second\_num, mode=None):  
 self.mode = mode # Сложение или вычитание  
 self.first\_num = first\_num # Первое число  
 self.second\_num = second\_num # Второе число  
 self.converted\_first = self.binarconverter(self.first\_num) # Первое в двоичном виде  
 self.converted\_second = self.binarconverter(self.second\_num) # Второе в двоичном виде  
 self.straight\_first = self.straight\_code(self.converted\_first, first\_num) # Первое в прямом коде  
 self.straight\_second = self.straight\_code(self.converted\_second, second\_num) # Второе в прямом коде  
 if int(first\_num) > 0: # Если первое положительное  
 self.reversed\_first = self.straight\_first # Обратный код равен прямому  
 self.additionally\_first = self.straight\_first # Дополнительный код равен прямому  
 else: # Иначе:  
 self.reversed\_first = self.reverse\_code(self.straight\_first) # Функция перевода в обратный код  
 self.additionally\_first = self.additionally\_code(str(self.first\_num)) # Функция перевода в доп.код  
 if int(second\_num) > 0: # Если второе положительое  
 self.reversed\_second = self.straight\_second # Обратный код равен прямому  
 self.additionally\_second = self.straight\_second # Дополнительный код равен прямому  
 else: # Иначе:  
 self.reversed\_second = self.reverse\_code(self.straight\_second) # Функция перевода в обратный код  
 self.additionally\_second = self.additionally\_code(str(self.second\_num)) # Функция перевода в доп.код  
 if self.mode == "+": # Если + то сложение  
 self.solution\_result = self.solution(self.additionally\_first, self.additionally\_second) # Функция сложения  
 elif self.mode == "-": # Если - вычитание  
 self.subtraction\_result = self.subtraction() # Функция вычитания  
  
 def binarconverter(self, x, res=''): # перевод десятичного числа в двоичную СС.  
 x = str(x)  
 if '-' in x:  
 x1 = '-'  
 x = int(x.replace('-', ''))  
 if x == 0:  
 return '0'  
 else:  
 while x > 0:  
 res = ('0' if x % 2 == 0 else '1') + res # Формула перевода в бинарный вид  
 x //= 2  
 return str(res)  
 else:  
 x = int(x)  
 if x == 0:  
 return '0'  
 else:  
 while x > 0:  
 res = ('0' if x % 2 == 0 else '1') + res # Формула перевода в бинарный вид  
 x //= 2  
 return str(res)  
  
 def straight\_code(self, num, ispositive):  
 l = len(num) # Длинна числа  
 if '-' in ispositive: # Если отрицательное:  
 num = '1' + ('0' \* (8 - l - 1)) + num  
 else: # Если положительное:  
 num = '0' + ('0' \* (8 - l - 1)) + num  
 return num  
  
 def reverse\_code(self, num):  
 new\_num = '' # Переменная для коррекции  
 num\_part = ''  
 if num[0] == '0': # Инверсия путем добавления в переменную для коррекции итоговых значений  
 num\_part = num[0] + ''  
 num = num[1:]  
 elif num[0] == '1':  
 num\_part = num[0] + ''  
 num = num[1:]  
 for i in range(len(num)):  
 if num[i] == '0':  
 new\_num += '1'  
 elif num[i] == '1':  
 new\_num += '0'  
 return num\_part + new\_num # Соединение итогового числа с разрядной цифрой  
  
 def additionally\_code(self, n, bits=8):  
 n = int(n)  
 mask = (1 << bits) - 1 # Побитовый снос  
 if n < 0:  
 n = ((abs(n) ^ mask) + 1)  
 answer = (self.binarconverter(n & mask)) # Перевод в 2 СС числа с бинарным операндом И  
 answer = str(answer)  
 return answer  
  
 def solution(self, num1, num2):  
 result = list() # Пустой список для записи результата сложения  
 p = 0 # Вспомогательная переменная для сложения (перенос единицы)  
 a = list(map(int, num1[:])) # Перевод всех чисел в списке в int  
 b = list(map(int, num2[:])) # Перевод всех чисел в списке в int  
  
 # Прохождение циклом в обратном порядке (от -1 до -7 (индексы в списке))  
 for i in range(-1, -8, -1):  
 if (a[i] + b[i] + p) == 0:  
 result.append('0')  
 p = 0  
 elif (a[i] + b[i] + p) == 1:  
 result.append('1')  
 p = 0  
 elif (a[i] + b[i] + p) == 2:  
 result.append('0')  
 p = 1  
 elif (a[i] + b[i] + p) == 3:  
 result.append('1')  
 p = 1  
 return "".join(result[::-1])  
  
 def subtraction(self):  
 ready\_digit = '-' + self.second\_num # Меняет знак второму числу на отрицательный  
 additionally\_digit = self.additionally\_code(ready\_digit) # Доп.код второго числа(уже отрицательного)  
 return self.solution(self.additionally\_first, additionally\_digit) # Сложение первого числа и отрицательного второго  
  
  
alphabet = '0123456789'  
a = input("Enter first digit:\n")  
b = input("Enter second digit:\n")  
mode = input("Enter mode(+,-):\n")  
for i in a:  
 if i in alphabet:  
 Flag = True  
 else:  
 Flag = False  
for j in b:  
 if j in alphabet:  
 Flag = True  
 else:  
 Flag = False  
if Flag:  
 ex = Digits(a, b, mode) # Инициализирую экземпляр класса(первое и второе число)  
 try:  
 if mode == '+':  
 print(ex.solution\_result) # Вывожу результат  
 elif mode == '-':  
 print(ex.subtraction\_result)  
 else:  
 print("Unknown mode.")  
 except AttributeError:  
 print("Error")  
else:  
 print("Error")

# Скріншоти:

