**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

Кафедра вычислительной техники

Отчет по лабораторной работе № 1

по дисциплине «Информатика»

Тема: **Системы счисления**

|  |  |
| --- | --- |
| Студент(ка) гр. 1305 | Смирнов М.О. |
| Студент гр. 1305 | Смирнов В.А. |
| Студент гр. 1305 | Чибисов А.А. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Перязева Ю.В. |

**Введение**

Цель работы состоит в закреплении знаний основных принципов представления чисел в различных системах счисления, которые необходимы для понимания того, как числовые данные хранятся и обрабатываются в компьютере.

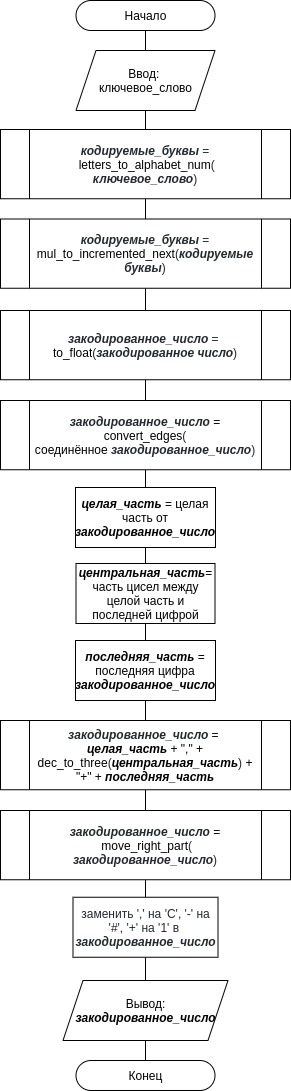
Для достижения цели необходимо выполнить следующие задачи:

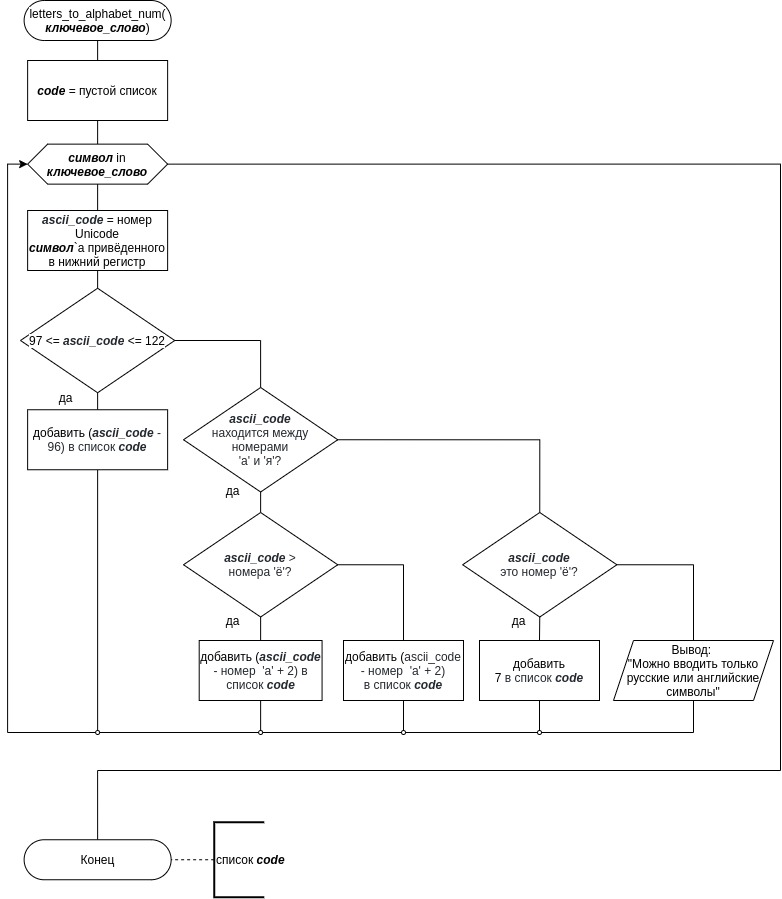
* изучить учебные материалы, посвященные системам счисления и хранению числовых данных в компьютере;
* разработать алгоритм по предложенному заданию, оформить в текстовом или графическом виде;
* разработать программу на языке программирования Python реализующую разработанный алгоритм;
* разработать контрольные примеры, выполнить их решение с помощью программы и ручной расчет, отладить программу;

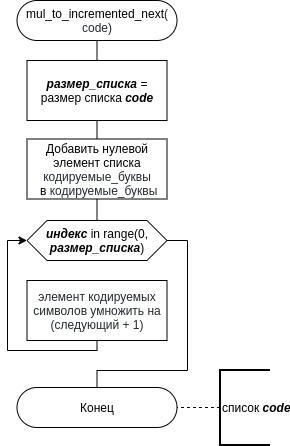
**Постановка задачи и описание решения**

Необходимо разработать алгоритм формирования кода для кодового замка, который состоит из n символов (цифры, латинские буквы и символ #) по слову-ключу. Разработка алгоритма должна быть основана на теоретических положениях представления чисел в различных системах счисления.

В ходе выполнения работы был разработан следующий алгоритм.





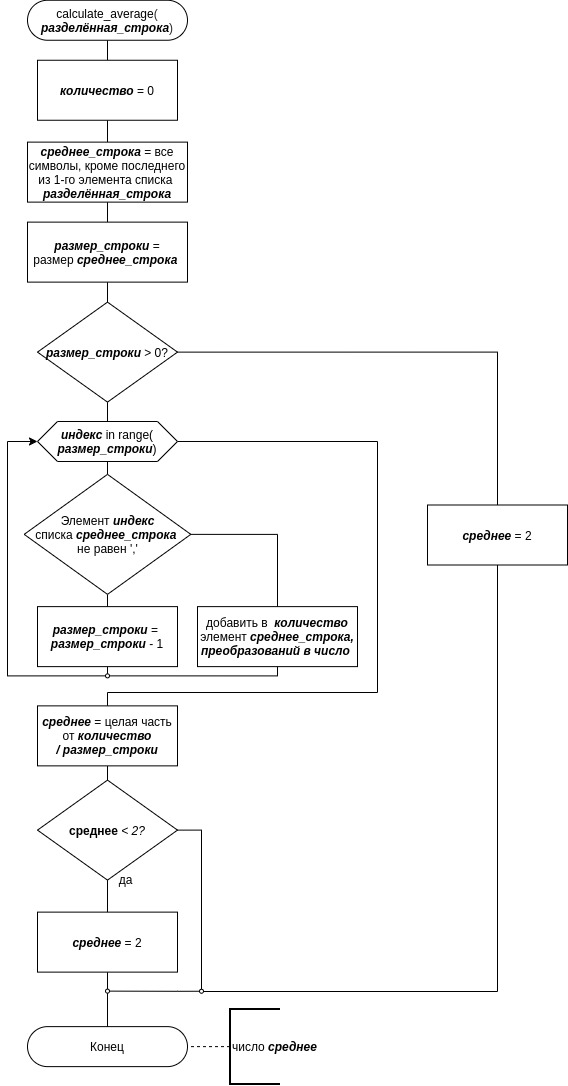


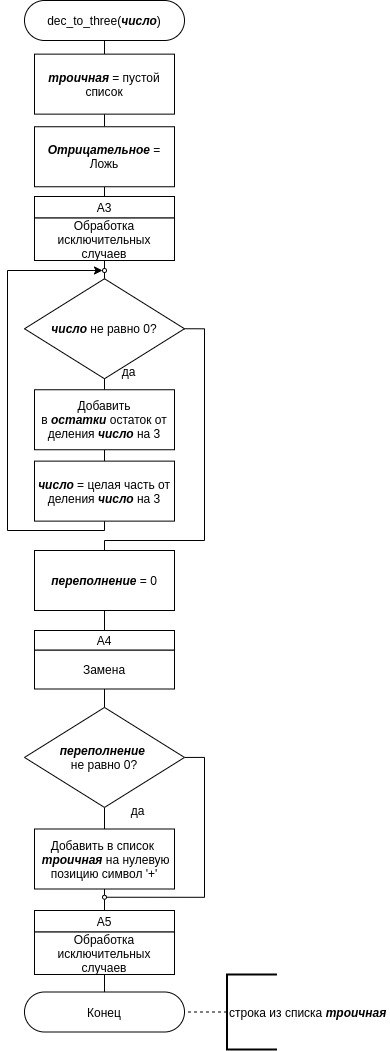
## 

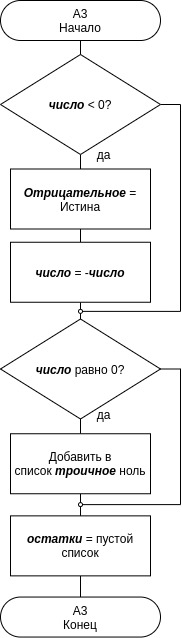
## 

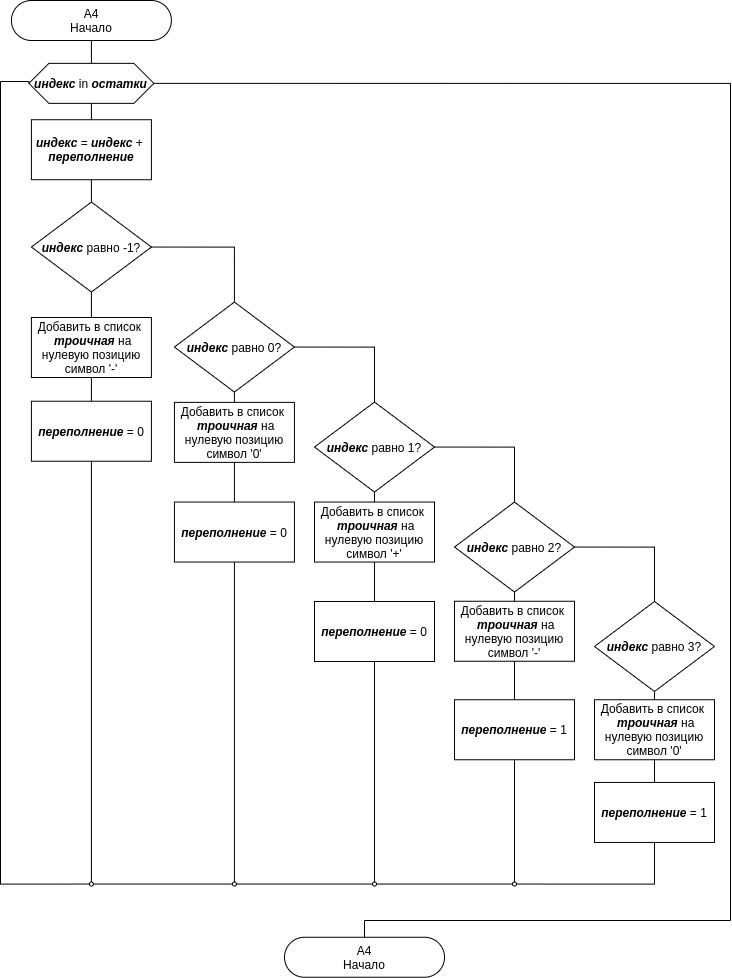
## 

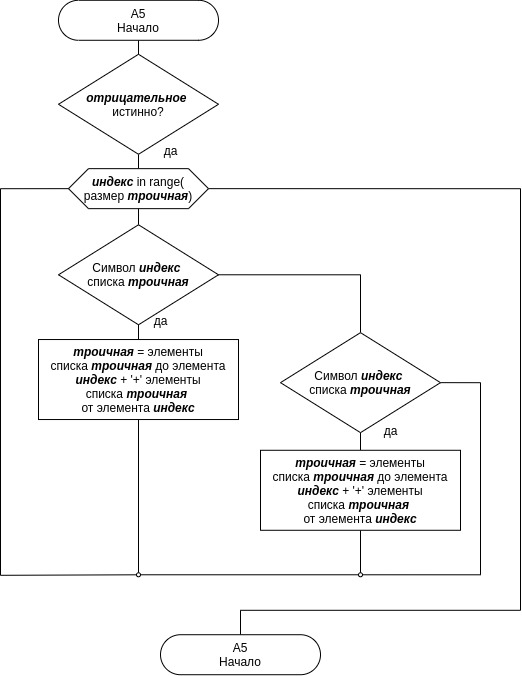
## 

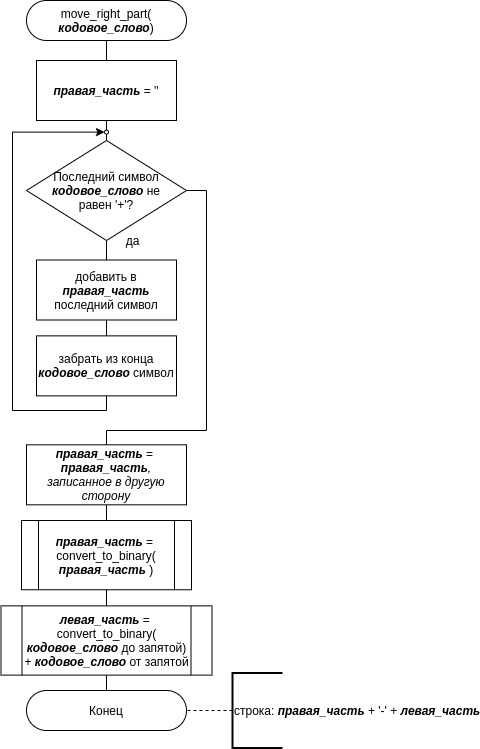


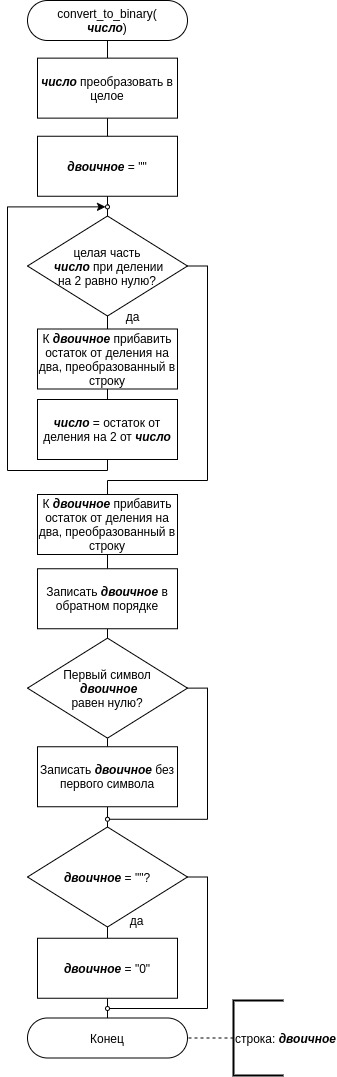












## Примеры работы программы:

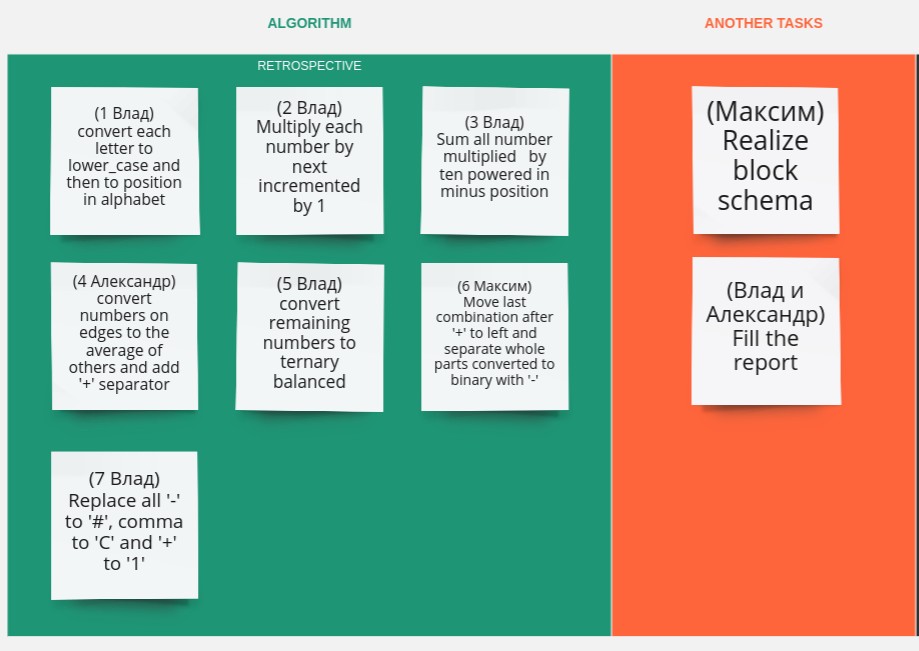
1) sUm => 0#11000101011101101C01

2) retroSpeCtIVe => 1100#1101110C10#101###11111#110##011##1

3) Мёд => 101#1110C10##01

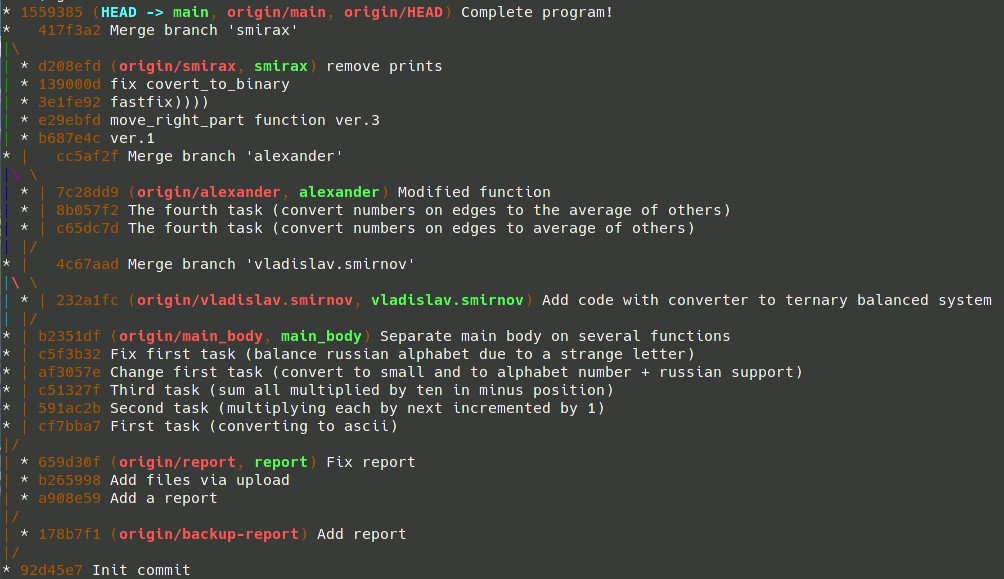
4) КалейДоскоП => 1#10C1##1111100###100#010001

Алгоритм реализован на Python 3, код представлен в приложении 1.

Скриншот данной kanban доски демонстрирует какие задачи были поставлены в ходе разработки алгоритма, а также подписаны исполнители.

## Разработка велась при помощи системы контроля версий git, а исходный код лежит на github по следующему адресу: <https://github.com/InformaticsLaboratory/Lab>.

## История разработки показана на данном скриншоте:

**Заключение**

Разработан алгоритм формирования кода для кодового замка, состоящего из n символов (цифры, латинские буквы и символ #), по слову ключу. Разработка алгоритма основана на теоретических положениях представления чисел в различных системах счисления. При разработке алгоритма было использовано два изученных алгоритма и один новый — перевод в троичную уравновешенную систему. Алгоритм оформлен графически (блоксхемы).

Были выполнены следующие задачи:

* Изучение учебных материалов, посвящённых системам счисления и хранению числовых данных в компьютере;
* Разработка алгоритма к заданию, оформление в графическом виде;
* Разработка программы на языке программирования Python, реализующей разработанный алгоритм;
* Разработка контрольных примеров и их решение с помощью программы и ручного расчёта, а также отладка программы;
* Подготовка отчёта.

В ходе работы были закреплены знания основных принципов

представления чисел в различных системах счисления, которые необходимы для понимания того, как числовые данные хранятся и обрабатываются в компьютере.

**Список использованных источников**

## Идея для перевода десятичного числа в троичную сбалансированную систему взята из данного источника: https://qna.habr.com/q/378857.

**Приложение 1**

def letters\_to\_alphabet\_num(keyword):

code = list()

for symbol in keyword:

ascii\_code = ord(symbol.lower())

if 97 <= ascii\_code <= 122:

code.append(ascii\_code - 96)

elif ord('а') <= ascii\_code <= ord('я'):

if ascii\_code > ord('е'):

code.append(ascii\_code - ord('а') + 2)

else:

code.append(ascii\_code - ord('а') + 1)

elif ascii\_code == ord('ё'):

code.append(7)

else:

print("You can input only russian or english letters!")

exit(0)

return code

def mul\_to\_incremented\_next(code):

list\_size = len(code)

coded\_letters.append(coded\_letters[0])

for index in range(0, list\_size):

coded\_letters[index] = coded\_letters[index] \* (coded\_letters[index + 1] + 1)

return coded\_letters[0:list\_size]

def to\_float(code):

list\_size = len(code)

float\_number = 0

for index in range(0, list\_size):

code[index] = str(code[index]) + "0" \* (list\_size - index - 1)

float\_number += int(code[index])

float\_number = list(str(float\_number))

encoded\_number\_size = len(float\_number)

if not list\_size <= 0:

if encoded\_number\_size > list\_size:

float\_number.insert(encoded\_number\_size - list\_size, ",")

elif encoded\_number\_size < list\_size:

for i in range(0, list\_size - encoded\_number\_size):

float\_number.insert(0, "0")

float\_number.insert(0, ",")

float\_number.insert(0, "0")

else:

float\_number.insert(0, ",")

float\_number.insert(0, "0")

return float\_number

def convert\_edges(string):

average = 0

spl\_str = []

new\_x = ''

new\_y = ''

if not(',' in string):

string += ',0'

spl\_str = string.split(',')

average = calculate\_average(spl\_str)

x = int(spl\_str[0])

y = int(spl\_str[1][-1:-2:-1])

while x != 0:

new\_x += str(x % average)

x //= average

while y != 0:

new\_y += str(y % average)

y //= average

if len(new\_x) == 0:

new\_x = '0'

if len(new\_y) == 0:

new\_y = '0'

new\_x = new\_x[::-1]

new\_y = new\_y[::-1]

new\_string = new\_x + ',' + spl\_str[1]

new\_string = new\_string[0:-1] + '+' + new\_y

return new\_string

def calculate\_average(spl\_str):

amount = 0

average\_str = spl\_str[1][0:-1]

len\_str = len(average\_str)

if len\_str > 0:

for i in range(len\_str):

if average\_str[i] != ',':

amount += int(average\_str[i])

else:

len\_str -= 1

average = int(amount // len\_str)

if average < 2:

return 2

else:

return 2

return average

def dec\_to\_three(n):

orientated = list()

is\_negative = False

if n < 0:

is\_negative = True

n = -n

if n == 0:

orientated.append(0)

remainders = list()

while n != 0:

remainders.append(n % 3)

n //= 3

overflow = 0

for i in remainders:

i = i + overflow

if i == -1:

orientated.insert(0, '-')

overflow = 0

elif i == 0:

orientated.insert(0, 0)

overflow = 0

elif i == 1:

orientated.insert(0, '+')

overflow = 0

elif i == 2:

orientated.insert(0, '-')

overflow = 1

elif i == 3:

orientated.insert(0, 0)

overflow = 1

if overflow != 0:

orientated.insert(0, '+')

if is\_negative:

for i in range(0, len(orientated)):

if orientated[i] == '-':

orientated = orientated[0:i] + list('+') + orientated[i + 1:]

elif orientated[i] == '+':

orientated = orientated[0:i] + list('-') + orientated[i + 1:]

return "".join(map(str, orientated))

def move\_right\_part(code\_word):

code\_word = str(code\_word)

right\_part = ""

while code\_word[-1] != "+":

right\_part += code\_word[-1]

code\_word = code\_word[:-1]

right\_part = right\_part[::-1]

right\_part = convert\_to\_binary(right\_part)

left\_part = convert\_to\_binary(code\_word[:code\_word.find(',')]) + (code\_word[code\_word.find(','):])

return right\_part + '-' + left\_part

def convert\_to\_binary(number):

number = int(number)

binary = ""

while number // 2 != 0:

binary += str(number % 2)

number //= 2

binary += str(number % 2)

binary = binary[::-1]

if binary[0] == '0':

binary = binary[1:]

if binary == '':

binary = "0"

return binary

print("Enter the keyword: ", end='')

user\_keyword = input()

coded\_letters = letters\_to\_alphabet\_num(user\_keyword)

coded\_letters = mul\_to\_incremented\_next(coded\_letters)

encoded\_number = to\_float(coded\_letters)

encoded\_number = convert\_edges(''.join(encoded\_number))

whole\_part = encoded\_number.split(',')[0]

middle\_part = encoded\_number.split(',')[1].split('+')[0]

end\_part = encoded\_number.split(',')[1].split('+')[1]

encoded\_number = whole\_part + "," + dec\_to\_three(int(middle\_part)) + "+" + end\_part

encoded\_number = move\_right\_part(encoded\_number)

encoded\_number = encoded\_number.replace(',', 'C').replace('-', '#').replace('+', '1')

print("The result code: " + encoded\_number)