```python
#Trabalho Final GCC-108 - Teoria da Computação
#Prof.: Douglas H. S. Abreu
#Nome: Deyvid Andrade Silva, 201820386
#Turma: 14A
#Link do repositório:
https://github.com/deyvidandrades/TrabalhoFinalGCC-208

#Definição da classe Fita:

class Fita:
    def __init__(self, configuracao: []) -> None:
        super().__init__()
        self.__config = configuracao
        self.__cabeca = 0

    @property
    def simbolo(self) -> str:
        if self.__cabeca < 0:
            lista = ['111']
            lista.extend(self.__config)
            self.__config = lista
            self.__cabeca = 0

        return self.__config[self.__cabeca]

    @property
    def cabeca(self) -> int:
        """
        :return: Posição da cabeça de leitura
        """
        return self.__cabeca

    @property
    def config(self) -> str:
        """
        Função para retornar a configuração da fita_3 ao fim da
execução da R(M) em U.
        :return: valor calulado na fita_3
        """
        return '#'.join(self.__config).replace('111', '').replace('#',
'')

    def escrever_simbolo(self, simbolo: str, direcao: str):
        """
        Escreve o símbolo de na posição atual e move a cabeça para a
próxima posição.
        :param simbolo: Simbolo para ser escrito
        :param direcao: esq ou dir
        """
        self.__config[self.__cabeca] = simbolo
```

```
        if direcao == 'esq':
            self.__cabeca -= 1
        else:
            self.__cabeca += 1

        if self.__cabeca >= len(self.__config):
            self.__config.append(' ')

    def __str__(self):
        return str({
            'cabeça': self.__cabeca,
            'configuração': self.__config
        })

# Funções auxiliares:

def binario_para_unario(valor: str) -> str:
    """
    Funcao para traduzir a entrada para unário
    :param valor: numero em binário
    :return: string em unário
    """

    valor = valor.replace("1", "0*")
    while "*0" in valor:
        valor = valor.replace("*0", "0**")

    valor = valor.replace("0", "").replace('*', '1')

    unario = ''
    for i in range(len(valor)):
        unario += '110'

    return unario


def unario_para_binario(valor: str) -> str:
    """
    Funcao para traduzir a saída para binário
    :param valor: numero em unário
    :return: string em binário
    """
    return bin(len(valor) - 1).replace("0b", "")

#Função principal:

def mtu(rep: str) -> str:
    """
    Função que simula a execução da MTU.
```

```
    :param rep: R(M)
    :return: valor calculado que foi armazenado na fita_3
    """

    # Separo a entrada em representação e palavra.
    m, w = rep.split('000')

    # Inicio as fitas com suas respectivas configurações (fita_1:
estados de R(M),
    # fita_2:  inicia com 1(zero em unário), fita_3: w)
    fita_1 = Fita(m.split('00'))
    fita_2 = Fita(['1'])
    fita_3 = Fita(w.split('0'))

    # loop principal, para quando o símbolo lido na fita_3 é
branco(posição vazia na fita).
    while fita_3.simbolo != ' ':
        x = fita_3.simbolo
        qi = fita_2.simbolo

        comparacao = f'{qi}0{x}0'

        # Retornando a cabeça da fita_1 para o inicio da fita.
        while fita_1.cabeca > 0:
            fita_1.escrever_simbolo(fita_1.simbolo, 'esq')

        # Selecionando a posição atual da fita_1 e fazendo o loop que
procura a transição correta para ser executada.
        transicao = fita_1.simbolo
        while transicao[:len(comparacao)] != comparacao:

            # Condição: quando as transições acabam, para e retorna o
resultado.
            if fita_1.simbolo == ' ':
                return fita_3.config, [str(fita_1), str(fita_2),
str(fita_3)]

            fita_1.escrever_simbolo(transicao, 'dir')
            transicao = fita_1.simbolo

        transicao = transicao.split('0')

        # Altera a fita_2 para o próximo estado a ser executado e
retorna para a primeira posição.
        fita_2.escrever_simbolo(transicao[2], 'dir')
        fita_2.escrever_simbolo(' ', 'esq')

        # Escreve o valor da transição atual na fita três e muda a
posição para a posição indicada pela transição.
```

```
        fita_3.escrever_simbolo(transicao[3], 'dir' if transicao[4] ==
'11' else 'esq')

    return fita_3.config, [str(fita_1), str(fita_2), str(fita_3)]

#Inicio da execução:

# Le o arquivo...
with open('entrada.csv', 'r') as file:
    data = file.readline()

# separa os números
num_1, num_2 = data.split(';')

# prepara a entrada
entrada =
f'{binario_para_unario(num_1)}1110{binario_para_unario(num_2)}111'

# adiciona a entrada no final da representação separadas por 000
representacao_soma_binaria =
f'10101010110010110101101100101110110111011001101011010110011011011011
01100110111011101110100111010111011010011101101111010100111011101111111
01110110011110101111010100111101111011010011110111011111011101001111
11010101101100111110110111110101001111101110101101100111111011011111110
11011001111110111011111111011101001111111011011111111011101001111111011
01111111101110100111111110101111111101010011111111011011111111101101000
{entrada}'

# executa a MTU para R(M)w
res, fitas = mtu(representacao_soma_binaria)

# Exibe os resultados
index = 1
print('Configurações:')
for f in fitas:
    print(f'fita_{index}: {f}\n')
    index += 1

print(f'\nResultados: ({num_1}+{num_2})')
print(f'resultado_unario={res}')
print(f'resultado_binario={unario_para_binario(res)}')
print(f'resultado_decimal={int(unario_para_binario(res), 2)}')

Configurações:
fita_1: {'cabeça': 21, 'configuração': ['1010101011', '101101011011',
'101110110111011', '110101101011', '11011011011011',
'1101110111011101', '11101011101101', '111011011110101',
'11101110111111110111011', '111101011110101', '11110110111101101',
'11101110111111011101', '111110101011011', '11111011011110101',
'11111011101011011', '1111110110111111011011',
```

'1111110111011111111011101', '1111111011011111111011101',
'11111110111011111111011101', '1111111101011111111010101',
'11111111011011111111101101', ' ']}

fita_2: {'cabeça': 0, 'configuração': ['11111111', ' ']}

fita_3: {'cabeça': 0, 'configuração': ['111', '11', '1', '1', '1',
'1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
'1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
'1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
'1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
'1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1',
'1', '1', '11', '11', '11', '11', '11', '11', '11', '11', '11', '11',
'11', '11', '1', '111', '111', '111', '111', '111', '111', '111',
'111', '111', '111', '111', '111', '111', '111', '111']}

Resultados: (1011000+1101)
resultado_unario=11111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111
resultado_binario=1100101
resultado_decimal=101