KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:

IFF-6/6 gr. studentas
Ignas Jasonas
2019 m. gegužės 27 d.

Priėmė:

Doc. Aštrys Kirvaitis

KAUNAS 2019

TURINYS

1. **Paveikslėlių sąrašas 3**
2. **Python (L1) 4**

2.1. Darbo užduotis 4

2.2. Programos tekstas 5

2.3. Pradiniai duomenys ir rezultatatai 6

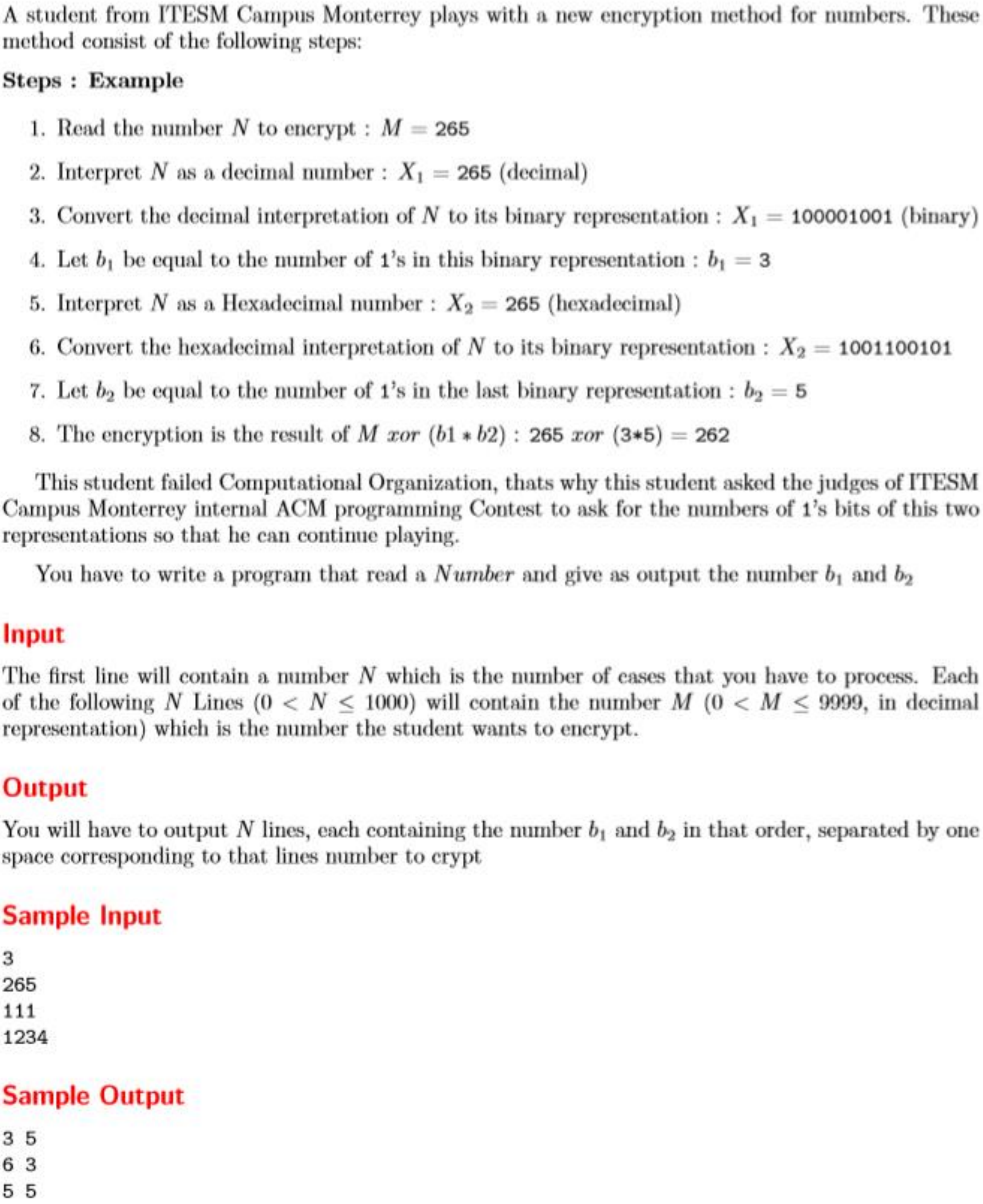
1. Paveikslėlių sąrašas

Pav. 1. Pasirinkta užduotis 4
Pav. 2. Programos kodas 5
Pav. 3. Duomenys 6
Pav. 4. Rezultatai 6

2. Python (L1)

2.1. Darbo užduotis

Pasirinkta užduotis:



A student from ITESM Campus Monterrey plays with a new encryption method for numbers. These method consist of the following steps:

Steps : Example

1. Read the number N to encrypt : $M = 265$
2. Interpret N as a decimal number : $X_1 = 265$ (decimal)
3. Convert the decimal interpretation of N to its binary representation : $X_1 = 100001001$ (binary)
4. Let b_1 be equal to the number of 1's in this binary representation : $b_1 = 3$
5. Interpret N as a Hexadecimal number : $X_2 = 265$ (hexadecimal)
6. Convert the hexadecimal interpretation of N to its binary representation : $X_2 = 1001100101$
7. Let b_2 be equal to the number of 1's in the last binary representation : $b_2 = 5$
8. The encryption is the result of M xor $(b1 * b2)$: 265 xor (3*5) = 262

This student failed Computational Organization, thats why this student asked the judges of ITESM Campus Monterrey internal ACM programming Contest to ask for the numbers of 1's bits of this two representations so that he can continue playing.

You have to write a program that read a $Number$ and give as output the number b_1 and b_2

Input

The first line will contain a number N which is the number of cases that you have to process. Each of the following N Lines ($0 < N \leq 1000$) will contain the number M ($0 < M \leq 9999$, in decimal representation) which is the number the student wants to encrypt.

Output

You will have to output N lines, each containing the number b_1 and b_2 in that order, separated by one space corresponding to that lines number to crypt

Sample Input

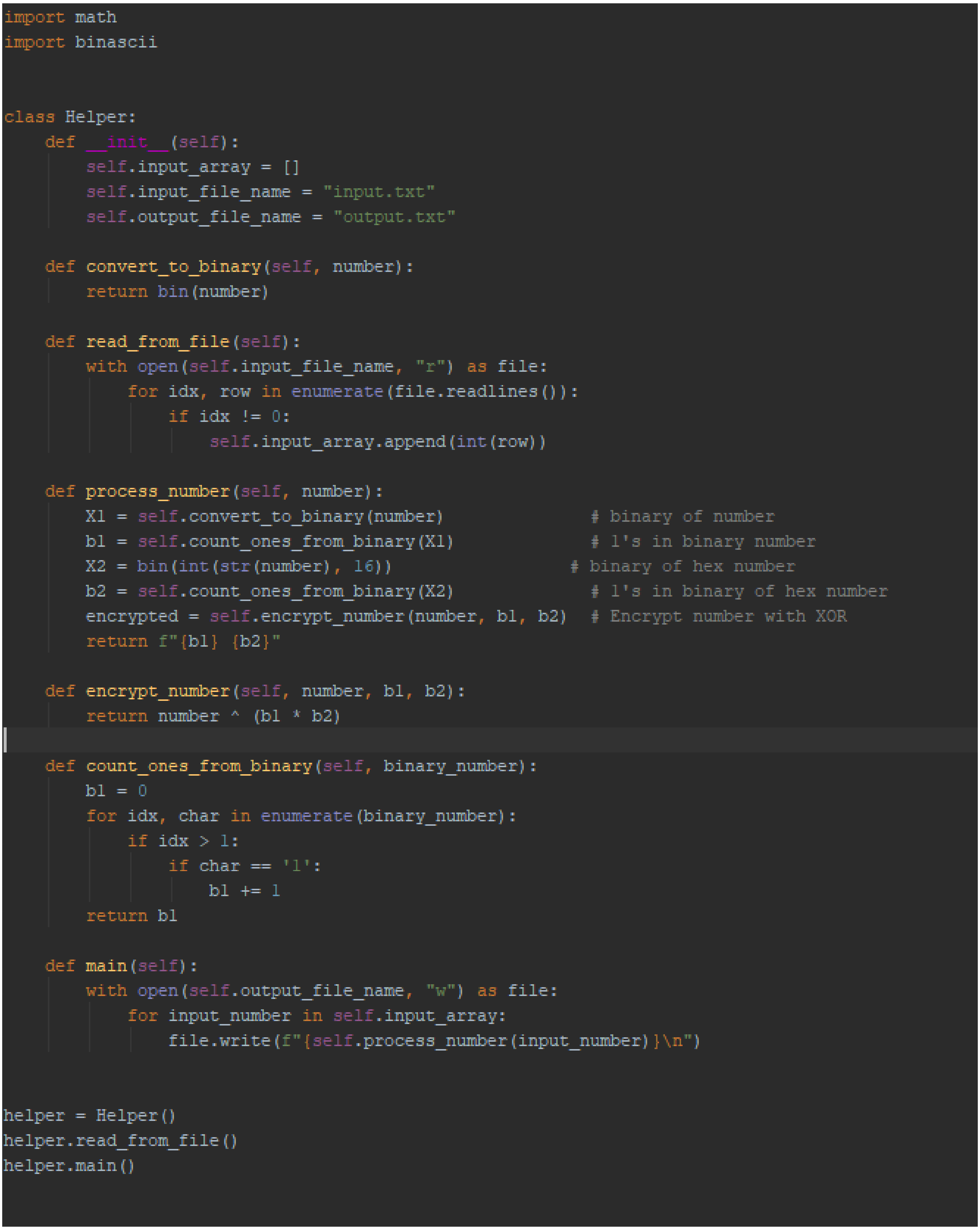
```
3
265
111
1234
```

Sample Output

```
3 5
6 3
5 5
```

Pav. 1. Pasirinkta užduotis

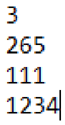
2.2. Programos tekstas



```python
import math
import binascii


class Helper:
    def __init__(self):
        self.input_array = []
        self.input_file_name = "input.txt"
        self.output_file_name = "output.txt"

    def convert_to_binary(self, number):
        return bin(number)

    def read_from_file(self):
        with open(self.input_file_name, "r") as file:
            for idx, row in enumerate(file.readlines()):
                if idx != 0:
                    self.input_array.append(int(row))

    def process_number(self, number):
        X1 = self.convert_to_binary(number)              # binary of number
        b1 = self.count_ones_from_binary(X1)             # 1's in binary number
        X2 = bin(int(str(number), 16))                  # binary of hex number
        b2 = self.count_ones_from_binary(X2)             # 1's in binary of hex number
        encrypted = self.encrypt_number(number, b1, b2)  # Encrypt number with XOR
        return f"{b1} {b2}"

    def encrypt_number(self, number, b1, b2):
        return number ^ (b1 * b2)

    def count_ones_from_binary(self, binary_number):
        b1 = 0
        for idx, char in enumerate(binary_number):
            if idx > 1:
                if char == '1':
                    b1 += 1
        return b1

    def main(self):
        with open(self.output_file_name, "w") as file:
            for input_number in self.input_array:
                file.write(f"{self.process_number(input_number)}\n")


helper = Helper()
helper.read_from_file()
helper.main()
```

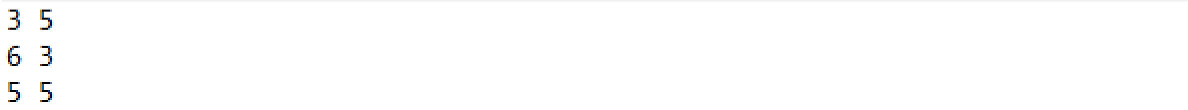
Pav. 2. Programos kodas

2.3. Pradiniai duomenys ir rezultatatai



```
3
265
111
1234
```

Pav. 3. Duomenys



```
3 5
6 3
5 5
```

Pav. 4. Rezultatai