Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 2

Дисципліна: Комп`ютерна логiка

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2018

**Лабараторна работа № 2**

1. **Розробити функцію** clean\_list(list\_to\_clean), яка приймає 1 аргумент -- список будь-яких значень (рядків, цілих та дійсних чисел) довільної довжини, та повертає список, який складається з тих самих значень, але не містить повторів елементів.

def clean\_list(list\_to\_clean):

temp = set(list\_to\_clean)

return temp

1. **Розробити функцію** counter(a, b), яка приймає 2 аргументи -- цілі невід'ємні числа a та b, та повертає число -- кількість різних цифр числа b, які містяться у числі а.

**def counter(a, b):**

**if a < 0 or b < 0:**

**return 0;**

**first\_arg = set(str(a));**

**second\_arg = set(str(b));**

**result = first\_arg.intersection(second\_arg);**

**return len(result);**

1. **Розробити функцію** super\_fibonacci(n, m), яка приймає 2 аргументи -- додатні цілі числа n та m (0 < n, m <= 35), та повертає число -- n-й елемент послідовності супер-Фібоначчі порядку m.

**def super\_fibonacci(n, m):**

**if n <= 0 and m > 35:**

**return 0;**

**temp = []**

**while m != 0:**

**temp.append(1)**

**m -= 1**

**result = len(temp) - 1**

**while len(temp) < n:**

**temp.append(sum(temp[-1 - result:]))**

**return (temp[-1])**

1. **Розробити функцію** file\_search(folder, filename), яка приймає 2 аргументи -- список folder та рядок filename, та повертає рядок -- повний шлях до файлу або папки filename в структурі folder.

**def file\_search(folder, filename):**

**try:**

**path = folder[0] + "/"**

**if filename in folder:**

**return (path + filename)**

**for element in folder:**

**if isinstance(element, list) == True:**

**if len(element) > 1:**

**result = file\_search(element, filename)**

**if path != None:**

**path = path + result**

**return (path)**

**except TypeError:**

**return False**

1. **Розробити функцію** count\_holes(n), яка приймає 1 аргумент -- ціле число або рядок, який містить ціле число, та повертає ціле число -- кількість "отворів" у десятковому записі цього числа друкованими цифрами (вважати, що у "4" та у "0" по одному отвору), або рядок ERROR, якщо переданий аргумент не задовольняє вимогам: є дійсним або взагалі не числом.

**def count\_holes(n):**

**if isinstance(n, int) or isinstance(n, str):**

**resource = {'0':1, '6':1, '4':1, '9':1 ,'8':2 }**

**counter = 0;**

**for char in str(n):**

**if char in resource:**

**counter += resource.get(char);**

**return counter;**

**else:**

**return 'ERROR'**

1. **Розробити функцію** encode\_morze(text), яка приймає 1 аргумент -- рядок, та повертає рядок, який містить діаграму сигналу, що відповідає переданому тексту, закодованому міжнародним кодом Морзе для англійського алфавіту. Розділові та інші знаки, що не входять до латинського алфавіту, ігнорувати. Регістром літер нехтувати.

**def encode\_morze(text):**

**morse\_code = {"A": "^\_^^^", "F": "^\_^\_^^^\_^", "P": "^\_^^^\_^^^\_^",**

**"B": "^^^\_^\_^\_^", "G": "^^^\_^^^\_^", "X": "^^^\_^\_^\_^^^",**

**"C": "^^^\_^\_^^^\_^", "H": "^\_^\_^\_^", "T": "^^^",**

**"D": "^^^\_^\_^", "K": "^^^\_^\_^^^", "W": "^\_^^^\_^^^",**

**"E": "^", "L": "^\_^^^\_^\_^", "S": "^\_^\_^",**

**"I": "^\_^", "J": "^\_^^^\_^^^\_^^^", "O": "^^^\_^^^\_^^^",**

**"M": "^^^\_^^^", "N": "^^^\_^", "Y": "^^^\_^\_^^^\_^^^",**

**"Q": "^^^\_^^^\_^\_^^^", "R": "^\_^^^\_^", "Z": "^^^\_^^^\_^\_^" ,**

**"U": "^\_^\_^^^", "V": "^\_^\_^\_^^^"}**

**text = text.upper();**

**text = text.strip();**

**list\_ = [];**

**result = '';**

**for char in text:**

**if char in morse\_code:**

**list\_.append(morse\_code.get(char))**

**elif char == " ":**

**list\_.append("\_")**

**result = "\_\_\_".join(list\_)**

**return (result)**

1. **Розробити функцію** saddle\_point(matrix), яка приймає 1 аргумент -- прямокутну матрицю цілих чисел, задану у вигляді списка списків, та повертає тьюпл із координатами "сідлової точки" переданої матриці або логічну константу False, якщо такої точки не існує.

**def saddle\_point(matrix):**

**l\_min = None;**

**y = 0;**

**l\_index = None;**

**if len(matrix) == 1:**

**return (0, 0)**

**while len(matrix) > y:**

**for i in matrix:**

**l\_min = min(i)**

**l\_index = i.index(l\_min)**

**for j in matrix:**

**if l\_min > j[l\_index]:**

**return (matrix.index(i), l\_index)**

**return False**

**y += gt1**

1. **Розробити функцію** find\_most\_frequent(text), яка приймає 1 аргумент -- текст довільної довжини, який може містити літери латинського алфавіту, пробіли та розділові знаки (,.:;!?-); та повертає список слів (у нижньому регістрі), які зустрічаються в тексті найчастіше.

**def find\_most\_frequent(text):**

**text = text.replace('-', ' ');**

**lists = text.split(' ');**

**resource = {}**

**for item in lists:**

**if item in resource:**

**temp = {item: (resource.get(item)+1)};**

**resource.update(temp)**

**else:**

**temp = {item: 1};**

**resource.update(temp);**

**sorted\_list = sorted(resource.items(), key=lambda x: x[1], reverse=True)**

**return sorted\_list[0]**

1. **Розробити функцію** convert\_n\_to\_m(x, n, m), яка приймає 3 аргументи -- ціле число (в системі числення з основою n) або рядок x, що представляє таке число, та цілі числа n та m (1 <= n, m <= 36), та повертає рядок -- представлення числа х у системі числення m.

У випадку, якщо аргумент х не є числом або рядком, або не може бути представленням цілого невід'ємного числа в системі числення з основою n, повернути логічну константу False.

def convert\_n\_to\_m(x, n, m):

if not isinstance(x, (int, str, bool)):

return False

elif ' ' in str(x):

x = x.replace(' ', '')

elif x == '' or x == True or '-' in str(x):

return False

x = str(x).upper();

ret\_string = ''

def convert\_in\_10(x, y):

x = str(x);

z = 1

lett = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9,'A': 10, 'B': 11, 'C': 12, 'D': 13, 'E': 14,

'F': 15, 'G': 16, 'H': 17, 'I': 18, 'J': 19,

'K': 20, 'L': 21, 'M': 22, 'N': 23, 'O': 24,

'P': 25, 'Q': 26, 'R': 27, 'S': 28, 'T': 29,

'U': 30, 'V': 31, 'W': 32, 'X': 33, 'Y': 34, 'Z': 35}

rec = (int(lett.get((x[0]))))

while z < len(x):

rec = (rec \* y + (int(lett.get((x[z])))))

z += 1

return rec

def convert\_from\_10(x, y):

list\_10 = [];

fract = None;

s = ''

lett = {0: '0', 1: '1', 2: '2', 3: '3', 4: '4',

5: '5', 6: '6', 7: '7', 8: '8', 9: '9',

10: 'A', 11: 'B', 12: 'C', 13: 'D', 14: 'E',

15: 'F', 16: 'G', 17: 'H', 18: 'I', 19: 'J',

20: 'K', 21: 'L', 22: 'M', 23: 'N', 24: 'O',

25: 'P', 26: 'Q', 27: 'R', 28: 'S', 29: 'T',

30: 'U', 31: 'V', 32: 'W', 33: 'X', 34: 'Y', 35: 'Z'}

while x >= y:

fract = int(x % y)

x = int(x / y)

list\_10.append(fract)

if x < y:

list\_10.append(x);

list\_10.reverse()

for i in list\_10:

if i in lett:

s += lett.get(i)

return s

try:

while x.startswith('0'):

x = x[1:]

var\_in\_10 = convert\_in\_10(x, n)

if m == 1:

for i in range(var\_in\_10):

ret\_string += '0'

return ret\_string

if convert\_from\_10(var\_in\_10, n) == str(x):

return convert\_from\_10(var\_in\_10, m)

else:

return False

except IndexError:

return '0'

TEST

import func

# print("Задание № 1");

# test\_list =['qwe', 'reg', 'qwe', 'REG']

# result = func.clean\_list(test\_list);

# print("Результат задания № 1: " + ' ' + str(result));

# print("Задание № 2");

# a = int(1233211);

# b = int(12128);

# result = func.counter(a, b);

# print("Результат задания № 2 :" + ' ' + str(result));

# print("Задание № 3");

# n = int(9);

# m = int(3);

# result = func.super\_fibonacci(n, m);

# print("Результат задания № 3: " + ' ' + str(result));

# print("Задание № 4");

# folder = [ '/home', ['user1'], ['user2', ['my pictures'], ['desktop', 'not this', 'and not this', ['new folder', 'hereiam.py' ] ] ],

# 'work.ovpn', 'prometheus.7z', ['user3', ['temp'], ], 'hey.py'];

# filename = 'hereiam.py'

# result = func.file\_search(folder, filename);

# print("Результат задания № 4" + ' ' + str(result));

# print("Задание № 5");

# n = "906"

# result = func.count\_holes(n);

# print("Результат задания № 5" + ' ' + str(result));

# print("Задание № 6");

# text = 'SOS'

# result = func.encode\_morze(text);

# print("Результат задания № 6" + ' ' + str(result));

# print("Задание № 7");

# matrix = [[8,3,0,1,2,3,4,8,1,2,3],

# [3,2,1,2,3,9,4,7,9,2,3],

# [7,6,0,1,3,5,2,3,4,1,1]];

# result = func.saddle\_point(matrix);

# print("Результат задания № 7" + ' ' + str(result));

# print("Задание № 8");

# text = 'Hello,Hello, my dear!';

# text = "text for text text test and-ther"

# result = func.find\_most\_frequent(text);

# print("Результат задания № 8" + ' ' + str(result));

print("Задание № 9");

result = func.convert\_n\_to\_m('0123', 4, 1);

print("Результат задания № 9" + ' ' + str(result));