

Міністерство освіти і науки, молоді та спорту України
Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

з домашнє завдання №27.2 з дисципліни:

“Алгоритми та моделі обчислень”

Варіант: № 24.

Виконав:

ст. групи КІ-203

Ширий Богдан Ігорович

Перевірів:

ст. викладач кафедри ЕОМ

Козак Назар Богданович

ЗАВДАННЯ:

УМОВА:

Виконати домашнє завдання №27.1 повторно за допомогою мови Python використовуючи RxPY (реалізація ReactiveX для Python)

ВИБІР ВАРІАНТУ:

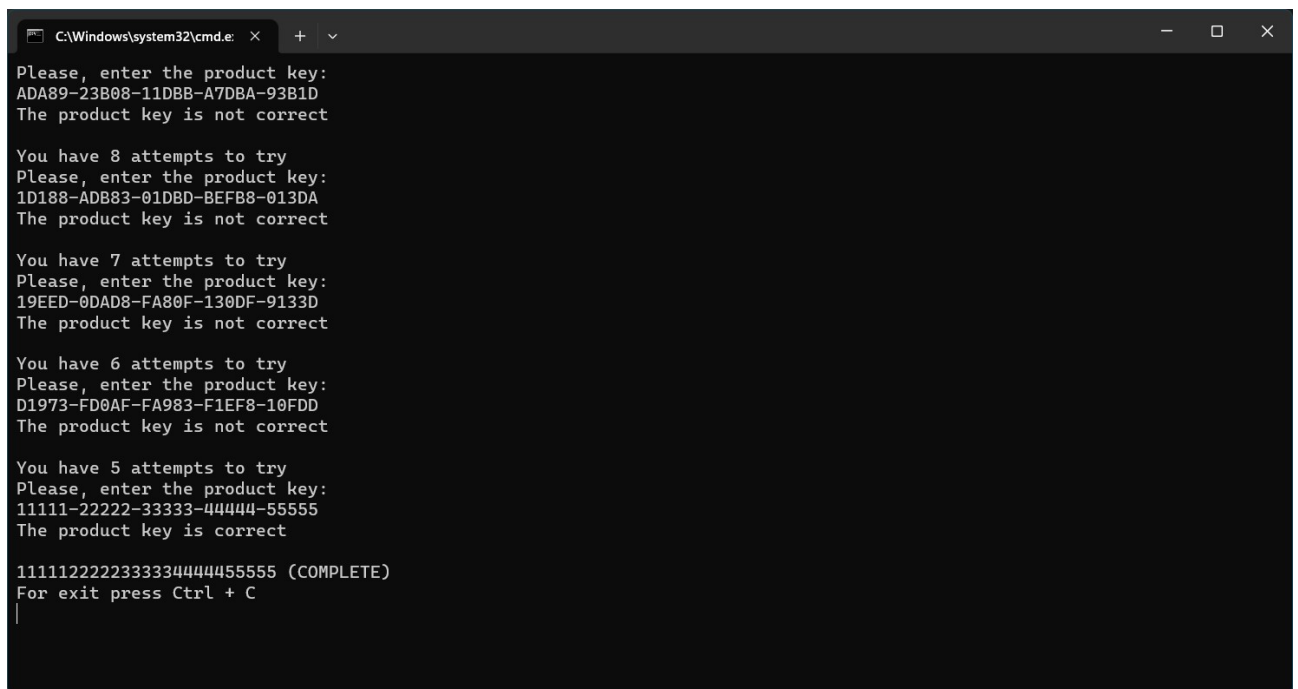
$$(N_{\text{ж}} + N_{\text{г}} + 1) \% 30 + 1 = (24 + 3 + 1) \% 10 + 1 = 28 \% 10 + 1 = 9,$$

де: $N_{\text{ж}}$ – порядковий номер студента в групі, а $N_{\text{г}}$ – номер групи.

Отож, мій шуканий варіант – це 9 можливих спроб введення ключа валідації.

ВИКОНАННЯ:

Склав Python програму та зобразив її роботу на рисунку 1.



```
C:\Windows\system32\cmd.e: x + v
Please, enter the product key:
ADA89-23B08-11DBB-A7DBA-93B1D
The product key is not correct

You have 8 attempts to try
Please, enter the product key:
1D188-ADB83-01DBD-BEFB8-013DA
The product key is not correct

You have 7 attempts to try
Please, enter the product key:
19EED-0DAD8-FA80F-130DF-9133D
The product key is not correct

You have 6 attempts to try
Please, enter the product key:
D1973-FD0AF-FA983-F1EF8-10FDD
The product key is not correct

You have 5 attempts to try
Please, enter the product key:
11111-22222-33333-44444-55555
The product key is correct

111112222233333444455555 (COMPLETE)
For exit press Ctrl + C
|
```

Рис. 1. Виконання програми написаної на Python.

Відповідно, у лістингу 1 навів код програми написаної на Python:

Лістинг 1. Код програми написаної на Python.

```
from rx.core import Observer
from rx import from_, operators as op
import curses, time
import re

ATTEMPTS_COUNT = 9
attemptsDownCount = ATTEMPTS_COUNT;

GROUPS_DIGITS_COUNT = 5;
```

```

GROUP_DIGITS_SIZE = 5;

PRODUCT_KEY_PART1 = [
0xF, 0xF, 0xF, 0xF, 0xF,
0xD, 0xD, 0xD, 0xD, 0xD,
0x8, 0x8, 0x8, 0x8, 0x8,
0xB, 0xB, 0xB, 0xB, 0xB,
0xF, 0xF, 0xF, 0xF, 0xF
];

PRODUCT_KEY_PART2 = [
0xE, 0xE, 0xE, 0xE, 0xE,
0xF, 0xF, 0xF, 0xF, 0xF,
0xB, 0xB, 0xB, 0xB, 0xB,
0xF, 0xF, 0xF, 0xF, 0xF,
0xA, 0xA, 0xA, 0xA, 0xA
];

DIGITS_COUNT = GROUPS_DIGITS_COUNT * GROUP_DIGITS_SIZE;

outOfEdgeIndex = 0;
currIndex = 0;
data = [0] * DIGITS_COUNT;

currRowIndex = 0;

def checkProductKey(productKey):
    for index in range(0, DIGITS_COUNT):
        if(productKey[index] ^ PRODUCT_KEY_PART1[index] ^ PRODUCT_KEY_PART2[index]):
            return False;
    return True;

def toDigitPosition(win, currRowIndex, currIndex):
    positionAddon = currIndex // GROUP_DIGITS_SIZE;
    if (positionAddon and positionAddon >= GROUPS_DIGITS_COUNT):
        positionAddon -= 1;
    win.move(currRowIndex, currIndex + positionAddon);
    pass;

def printProductKey(win, productKey, outOfEdgeIndex):
    global currIndex;
    global currRowIndex;
    global DIGITS_COUNT;
    win.move(currRowIndex, 0);
    for index in range(0, DIGITS_COUNT):
        if(index >= outOfEdgeIndex):
            break;
        win.addstr("{:X}".format(productKey[index]))
    pass;

def printFormattedProductKey(win, productKey, outOfEdgeIndex):
    global currIndex;
    global currRowIndex;
    global GROUP_DIGITS_SIZE;
    global DIGITS_COUNT;
    win.move(currRowIndex, 0);
    for index in range(0, DIGITS_COUNT):
        if(index >= outOfEdgeIndex):
            break;
        win.addstr("{:X}".format(productKey[index]))
        if(not((index + 1) % GROUP_DIGITS_SIZE) and (index + 1) < DIGITS_COUNT):
            win.addstr( '-' );
    pass;

def inputHandler(win, ch, key):
    global currIndex;
    global currRowIndex;
    global outOfEdgeIndex;

```

```

global attemptsDownCount;
if(not attemptsDownCount):
    return;
if key in (10, curses.KEY_ENTER):
    if (checkProductKey(data) ):
        toDigitPosition(win, currRowIndex, outOfEdgeIndex);
        win.addstr("\nThe product key is correct\n\n");
        currRowIndex += 3;
        toDigitPosition(win, currRowIndex, outOfEdgeIndex);
        printProductKey(win, data, outOfEdgeIndex)
        win.addstr(' (COMPLETE)\n');
        win.addstr('For exit press Ctrl + C\n');
        currRowIndex += 2;
        attemptsDownCount = 0;
    else:
        toDigitPosition(win, currRowIndex, outOfEdgeIndex);
        win.addstr("\nThe product key is not correct\n");
        currRowIndex += 2;
        attemptsDownCount-=1;
        win.addstr("\nYou have " + str(attemptsDownCount) + " attempts to try");
        currRowIndex += 1;
        if(attemptsDownCount):
            win.addstr("\nPlease, enter the product key:\n");
            currRowIndex += 2;
            printFormattedProductKey(win, data, outOfEdgeIndex);
            toDigitPosition(win, currRowIndex, currIndex);
        else:
            win.addstr('\nThe product key is not entered\n');
            win.addstr('For exit press Ctrl + C\n');
            currRowIndex += 3;
if key in (8, curses.KEY_BACKSPACE):
    if(currIndex):
        currIndex-=1;
        toDigitPosition(win, currRowIndex, currIndex);
        data[currIndex] = 0;
        win.addstr( '0' );
        toDigitPosition(win, currRowIndex, currIndex);
if key in (127, curses.KEY_DC):
    toDigitPosition(win, currRowIndex, currIndex);
    data[currIndex] = 0;
    win.addstr( '0' );
    toDigitPosition(win, currRowIndex, currIndex);
elif key in (27, curses.KEY_LEFT):
    if(currIndex):
        currIndex-=1;
        toDigitPosition(win, currRowIndex, currIndex);
elif key in (26, curses.KEY_RIGHT):
    if(currIndex < outOfEdgeIndex):
        currIndex+=1;
        toDigitPosition(win, currRowIndex, currIndex);

hexDigitRegularExpression = r'^[0-9A-Fa-f]\b';
re.match(hexDigitRegularExpression, ch)
if (ch and re.match(hexDigitRegularExpression, ch) and currIndex <
DIGITS_COUNT):
    data[currIndex] = int(ch, 16);
    win.addstr(ch.upper());
    if(outOfEdgeIndex <= currIndex):
        outOfEdgeIndex = currIndex + 1;
    if(currIndex + 1 < DIGITS_COUNT):
        currIndex+=1;
        if (currIndex != DIGITS_COUNT and currIndex % 5 == 0):
            win.addstr( '-' );
    if(currIndex + 1 == DIGITS_COUNT):
        toDigitPosition(win, currRowIndex, currIndex);
pass

win = curses.initscr()

```

```

curses.noecho()
win.keypad(1)
class ConsoleKeyInputDataSource:
    def __init__(self, win):
        self.win = win
    def __iter__(self):
        while True:
            try:
                time.sleep(0.01)
                yield win.getch()
            except KeyboardInterrupt:
                break;

class PrintObserver(Observer):
    global currRowIndex;
    def __init__(self, win):
        self.win = win;
    def on_next(self, value):
        inputHandler(win, chr(value), value)
    def on_completed(self):
        win.addstr("\nDone!\n")
    def on_error(self, error):
        win.addstr("\nError Occurred: {0}\n".format(error))

if(attemptsDownCount):
    win.addstr('Please, enter the product key:\n');
    currRowIndex +=1;

inputObservable = from_(ConsoleKeyInputDataSource(win));
inputObservable = inputObservable.pipe(
    op.map(lambda key :
        ord('0') if (key and (chr(key) == ' ' or chr(key) == '\t')) else key)
)
inputObservable.subscribe(PrintObserver(win));

```