אתגרי Check Point 2018

מאת Dvd848

מבוא

חברת צ'ק-פוינט פרסמה סדרה של אתגרים במסגרת מסע הפרסום של "האקדמיה הראשונה לסייבר מבית צ'ק-פוינט". האתגרים הגיעו ממספר תחומים, ביניהם Web, Reversing, Programming, Networking ו-Logic. במאמר זה אציג את הפתרונות שלי לאתגרים אלו.

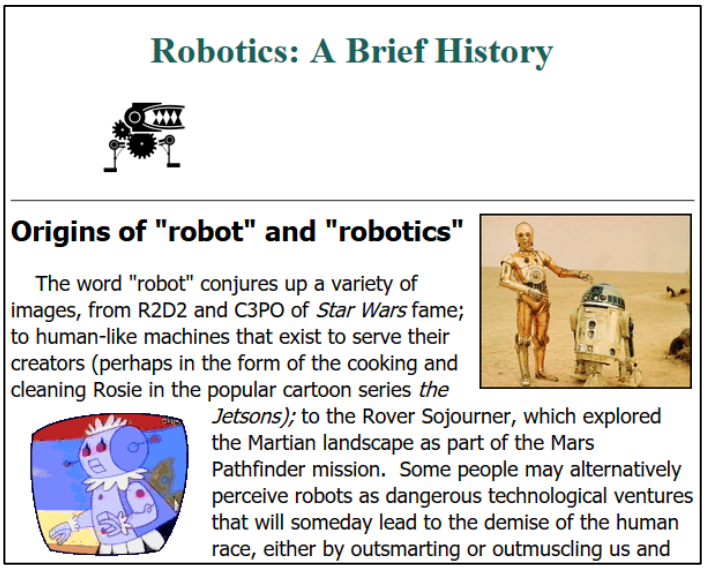
אתגר 1: Return of the Robots (קטגוריית Web, 10 נקודות)

הוראות האתגר:

Return of the Robots

Robots are cool, but trust me: their access should be limited!

לפסקה צורף קישור לאתר עם טקסט על היסטוריית הרובוטיקה:



Robotics: A Brief History

Origins of "robot" and "robotics"

The word "robot" conjures up a variety of images, from R2D2 and C3PO of *Star Wars* fame; to human-like machines that exist to serve their creators (perhaps in the form of the cooking and cleaning Rosie in the popular cartoon series *the Jetsons);* to the Rover Sojourner, which explored the Martian landscape as part of the Mars Pathfinder mission. Some people may alternatively perceive robots as dangerous technological ventures that will someday lead to the demise of the human race, either by outsmarting or outmuscling us and

מה שהטקסט נמנע מלהזכיר הוא כמובן שבעולם ה-Web, המונח Robots מיד מקפיץ אסוציאציה של הקובץ robots.txt, או בשמו הרשמי יותר "פרוטוקול אי הכללת רובוטים".

מדובר בפרוטוקול שמאפשר לבעלי אתרים לבקש מבוטים של מנועי חיפוש שסורקים את האינטרנט להימנע מלכלול דפים מסוימים של האתר בתוצאות מנוע החיפוש. כאשר מנוע החיפוש מגיע לאתר, הוא אמור לבדוק את התוכן של הקובץ robots.txt בתיקיית השורש של האתר. אם קובץ כזה קיים, מנוע החיפוש לא אמור לאנדקס כתובות שמצוינות בקובץ (כמובן שזוהי מוסכמה ושום דבר לא מונע ממנוע חיפוש לאנדקס מה שהוא רוצה, כל עוד יש לו גישה לדף).

כלומר, אם קיימים דפים שמנהל האתר לא מעוניין לחשוף באופן פומבי, הוא יכול לכלול אותם בקובץ הזה. אולם, זה מייצר בעיה אחרת, מעצם העובדה שהקובץ הזה חייב להיות פומבי: הוא כולל רשימה ממוקדת ונגישה של כל הדפים שאין להם עניין ציבורי.

אם ננסה לקרוא את הקובץ מהשרת של האתגר, נמצא את התוכן הבא:

```
User-agent: *
Disallow: /secret_login.html
```

ניגש לדף ונראה:

No Robots Allowed

Password: [ ] Submit

קוד המקור של הדף נראה כך:

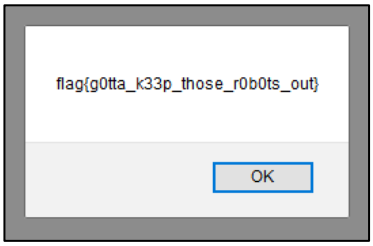
```
<html>
 <body>
  <script type="text/javascript">
  function r(n) {
    for (var r=0, o=0, e="", t=0; t<n.length; t++)
       n[t].toLowerCase() != n[t] && (r+=1), 8 == ++o
        ? (e += String.fromCharCode(r), r=0, o=0) : r <<= 1; alert(e)
  }

  function auth(n) {
    if ("SzMzcFQjM1IwYjB0JDB1dA==" == btoa(n))
    var a =
"pVPmwMHevTZoIGjevOQdfpiLwEQwxYINxOBVNyFGhUPimVXUhdMWqrzmjAXIzTpvlZFXgFvisSEnblc
PnLfZUBUPnZPtXwQOpnUWfyAUhbANrqOKySBErmflnHfWLVAXvOSKpCqwaWWvLrskwFNxWTYTnCAKteT
GjYIxsKpXwGuDNWXLyMTVphBuryEVylptvSDaxrMnmgPSokwcfDIVhNsutQCLppSVjYiQFLNWtCVeRRT
ZkRQEsMzDhBPMrSycaHGWMDpY";
    else a =
"sRnDjXnrzAZVoxXnjSWLUoyWtgQpzziflCuxapkGjYEcrUADyMZlgunEaXLqYncWlHGpIVMvltZxveo
E";
     r(a)
  }
  </script>
  <h1>No Robots Allowed</h1>
  <label for="userPassword">Password: </label>
  <input id="userPassword" type="password" required>
  <input type="submit" value="Submit" onclick = "auth(userPassword.value);">
 <body>
</html>
```

אפשר לראות שפונקציית auth משווה את הסיסמא שהתקבלה מהמשתמש אל ערך קבוע (מקודד ב-Base64, כפי שאפשר לראות בין השאר מהשימוש בפונקציית btoa שמקודדת מחרוזת ב-Base64).

נשתמש בפונקציית atob לפענוח הקידוד ונקבל את הסיסמא:

```
>> atob("SzMzcFQjM1IwYjB0JDB1dA==")
"K33pT#3R0b0t$0ut"
```

בתגובה, הדף יקפיץ את הדגל:



אתגר 2: Diego's Gallery (קטגוריית Web, 20 נקודות)

הוראות האתגר:

Diego's Gallery

Recently I've been developing a platform to manage my cat's photos and keep my flag.txt safe. Please check out my beta

To avoid security loop holes such as SQL injections I developed my own scheme.

Every line in my DB look's like this:

> START|||username|||password|||role|||END

So for example:

> START|||diego|||catnip|||admin|||END

> START|||joe|||1234567|||user|||END

האתר מכיל טופס התחברות פשוט:

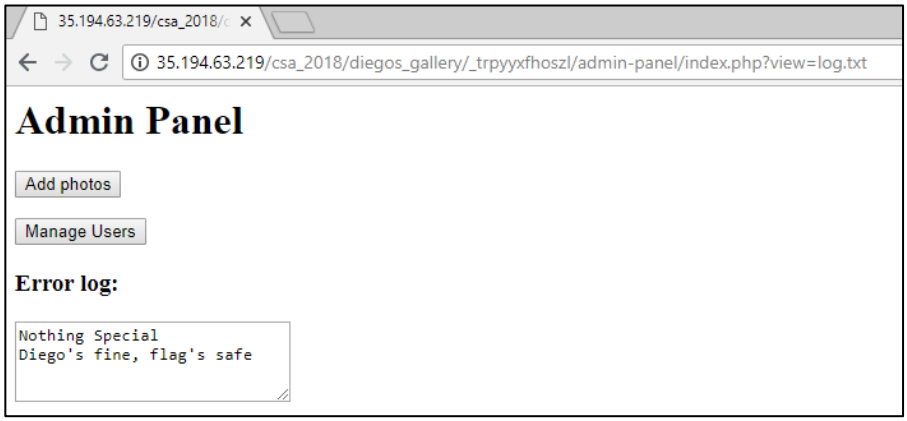
כמו ב-SQL Injection בסיסי, נרצה להכניס קלט באחד השדות שישפיע על התחביר במקום רק על הנתונים. למשל, אם במקום הסיסמא, נכניס:

```
some_password|||admin|||END
```

התחביר הסופי יהיה:

```
START|||some_username|||some_password|||admin|||END|||user|||END
```

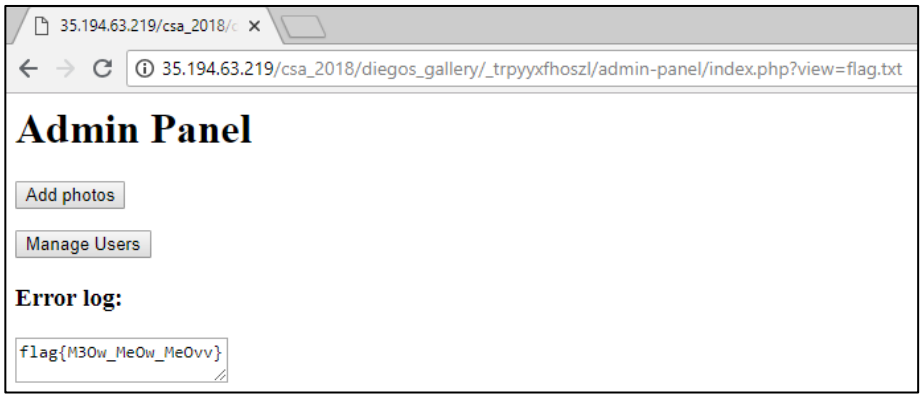
וכך נצליח לגרום לקוד לחשוב שהמשתמש שלנו הוא מנהל, ונקבל גישה לדף הניהול:



כפתורי הניהול לא עושים שום דבר מעניין, אך שימו לב לשורת הכתובות:

http://35.194.63.219/csa_2018/diegos_gallery/_trpyyxfhoszl/admin-panel/index.php?view=log.txt

הקובץ index.php מקבל כפרמטר שם של קובץ ומציג את התוכן שלו. מה יקרה אם במקום log.txt נבקש קובץ אחר, למשל flag.txt?



אתגר 3: Careful Steps (קטגוריית Programming, 20 נקודות)

הוראות האתגר:

> This is a bunch of archives we've found and we believe a secret flag is somehow hidden inside them.
>
> We're pretty sure the information we're looking for is in the comments section of each file.
>
> Can you step carefully between the files and get the flag?
>
> Good luck!

קובץ הארכיון מכיל 2000 קבצים, החל מ-unzipme.0 ועד ל-unzipme.1999. שימוש בפקודת file מראה שמדובר באוסף של קבצי RAR ו-ZIP:

```
root@kali:~/Downloads/checkpoint/archives# file unzipme.0
unzipme.0: RAR archive data, v4, os: Unix
root@kali:~/Downloads/checkpoint/archives# file unzipme.2
unzipme.2: Zip archive data, at least v2.0 to extract
root@kali:~/Downloads/checkpoint/archives#
```

התוכן לא נראה מעניין כל כך, אבל ההוראות שלחו אותנו להערות (שני הפורמטים תומכים בהוספת הערות לקובץ הארכיון).

אפשר לראות הערה של קובץ ZIP באמצעות פקודת unzip -z:

```
root@kali:~/Downloads/checkpoint/archives# unzip -z unzipme.2
Archive:  unzipme.2
p,-19
root@kali:~/Downloads/checkpoint/archives# unzip -z unzipme.4
Archive:  unzipme.4
Q,-68
root@kali:~/Downloads/checkpoint/archives# unzip -z unzipme.5
Archive:  unzipme.5
j,-10
root@kali:~/Downloads/checkpoint/archives#
```

נראה שכל הערה כוללת אות, ומספר. ננסה להתייחס אל המספר בתור הוראות לאיזה קובץ לקפוץ בצעד הבא, ואל האות בתור חלק מהדגל.

לשם כך נוכל להשתמש בסקריפט הבא:

```
import os
import zipfile
import rarfile
import sys


print ("Reading comments...")
listOfFiles = sorted(os.listdir('archives'))
comments = {}
```

```
for file_name in listOfFiles:
    try:
        with zipfile.ZipFile('archives/' + file_name) as zf:
            comment = zf.comment.decode("utf-8")
    except zipfile.BadZipFile:
        try:
            with rarfile.RarFile('archives/' + file_name) as rf:
                comment = rf.comment
        except e:
            raise e
    #print ("{}\t{}".format(file_name, comment))
    comments[int(file_name.replace("unzipme.", ""))] = comment.rstrip()

print ("Following trail...")
current_index = 0
new_offset = 0
while True:
    current_index = current_index + new_offset
    #print ("Trying to access {}".format(current_index + new_offset))

    current = comments[current_index]
    char, new_offset = current.split(",")
    new_offset = int(new_offset)
    #print ("{}, {}".format(char, new_offset))
    sys.stdout.write(char)

    if new_offset == 0:
        break
```

החלק הראשון קורא את כל ההערות, והחלק השני עוקב אחרי הצעדים בהערות ומדפיס את התווים המתאימים. אם נריץ את הסקריפט, נקבל:

```
Reading comments...
Following trail...
Well done buddy! You seem to be able to step carefully through the files. This i
s your flag: flag{ArchvIE$_Ar3_ThE_BeS7}
>>>
```

אתגר 4: Ping Pong (קטגוריית Networking, 25 נקודות)

הוראות האתגר:

I bet you're not fast enough to defeat me.

I'm at: nc 35.157.111.68 10140

נתחבר לשרת:

```
root@kali:~/Downloads/checkpoint# nc 35.157.111.68 10140
Welcome!
Send the following number: 991082
991082
Good, the next is: 708957
708957
Good, the next is: 856185
You're Too Slow For Me...
root@kali:~/Downloads/checkpoint#
```

השרת מבקש שנשלח לו מספר אקראי כלשהי. כאשר אנו עושים זאת, הוא מבקש מספר אחר. אם התגובה איטית מדי, השרת סוגר את החיבור. כמובן שאנחנו לא רוצים לשלוח תשובות ידנית ולכן נכתוב סקריפט שעושה זאת עבורנו:

```
import socket

import time
import re


s = socket.socket()

reg = re.compile('^.+: ([\d]+)\n$')

try:
    port = 10140

    s.connect(('35.157.111.68', port))

    start_time = time.time()
    print (s.recv(9)) #Read the "Welcome!\n"
    while True:
        msg = (s.recv(1024)).decode("utf-8")
        print (msg)
        match = reg.match(msg)
        if match:
            num = match.group(1)
            print (num)
            s.send(str.encode(num + "\n"))
        else:
            break
    print("--- %s seconds ---" % (time.time() - start_time))


except:
    raise
finally:
    s.close()
```

הסקריפט משתמש בביטוי רגולרי כדי לחלץ את המספר ואז שולח אותו חזרה אל השרת:

```
re.compile('^.+: ([\d]+)\n$')
```

הביטוי הזה מתאים לשורה שמתחילה בכל תו (.) שמופיע פעם אחת או יותר (+) כאשר לאחר מכן צריכות להופיע נקודתיים (:) ואז רווח ( ), ספרה אחת או יותר (+[\d]) וירידת שורה (\n). הסימנים "^" ו"$" מסמלים תחילת וסוף שורה, והסוגריים מסביב ל-"+[\d]" מסמנים שזהו הביטוי שנרצה לחלץ.

את הביטוי אנחנו מקמפלים מראש כדי להשיג ריצה יעילה יותר.

נריץ את הסקריפט ונקבל:

```
913794
Good, the next is: 224469

224469
Good, the next is: 49506

49506
Good, the next is: 858163

858163
flag{SEeMS_LiKE_YOu_StiLl_Fa$t}

--- 136.29361844062805 seconds ---
>>>
```

בדיעבד, מכיוון שאנחנו יודעים שהשרת תמיד מחזיר את אותה תשובה, היה אפשר לוותר על הביטוי הרגולרי ולחסוך כמה שניות (במחיר של קריאות וגמישות) על ידי דילוג על "Good, the next is: " וקפיצה ישירה אל המספר שצריך להחזיר (במילים אחרות, נראה שהמספר תמיד מתחיל באותו offset מתחילת השורה).

אתגר 5: Protocol (קטגוריית Networking, 30 נקודות)

הוראות האתגר:

Hi there!

We need to extract secret data from a special file server.

We don't have much details about this server, but we did manage to intercept traffic containing communication with the server.

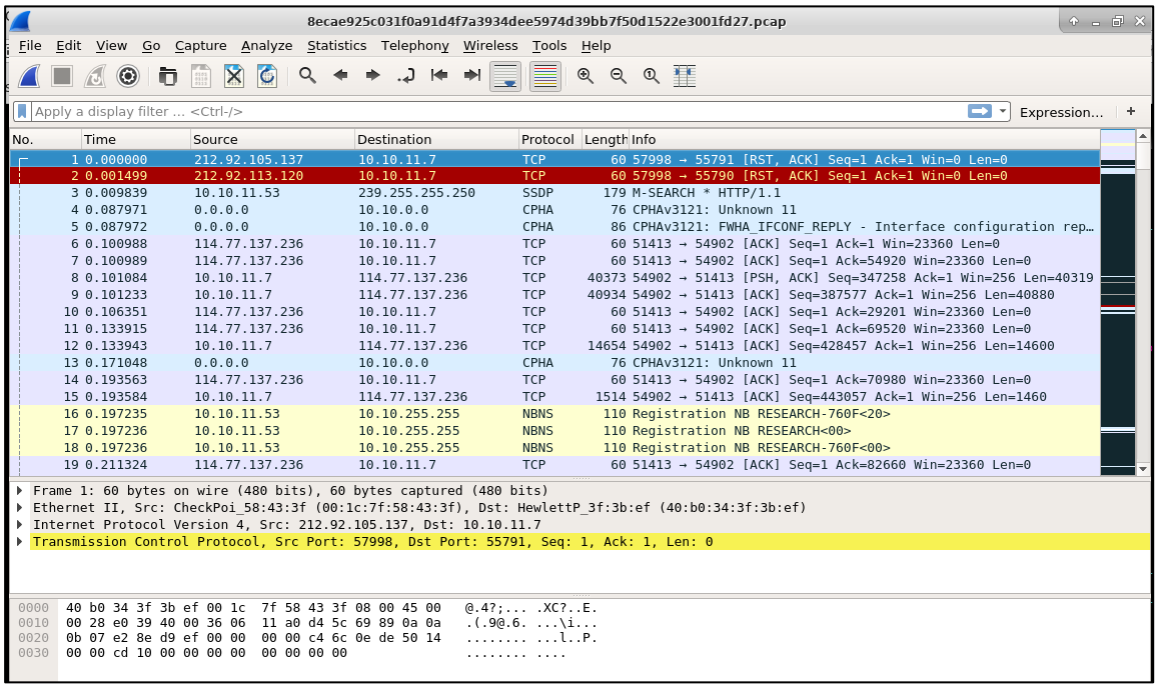
We also know that this secret file's path is: /usr/7Op_sECreT.txt

You can find the sniff file here.

Please tell us what the secret is!

Good luck!

הקובץ שמתקבל הוא קובץ pcap שמשמש להצגת תעבורת רשת וניתן לפתיחה באמצעות תוכנת WireShark.



מעבר זריז על ההודעות השונות ועיון ב-payload מגלה הודעה מעניינת:

```
0000  00 1c 7f 58 43 3f 40 b0  34 3f 3b ef 08 00 45 00   ...XC?@. 4?;...E.
0010  00 32 42 06 40 00 80 06  00 00 0a 0a 0b 07 23 9d   .2B.@... ......#.
0020  6f 44 d9 f1 4e 21 2b 7d  d6 ad 0d a2 27 ae 50 18   oD..N!+} ....'.P.
0030  01 00 a8 16 00 00 31 20  35 20 48 45 4c 4c 4f 0a   ......1  5 HELLO.
```

ה-HELLO קופץ מיד לעין. ניתן לעקוב אחרי כל ההודעות של החיבור הזה באמצעות קליק ימני ובחירה ב-Follow TCP Stream:

```
0 8 Welcome!
1 5 HELLO
1 5 HELLO
2 3 XOR
2 4 5C1A
3 12 /usr/bed.txt
3 264
0x117f0x7c7b0x327e0x7c770x253a0x3e7f0x383a0x3d680x393a0x2c7f0x2e7c0x39790x283
a0x3a750x2e3a0x397b0x3f720x7c750x28720x39680x703a0x3e6f0x283a0x31630x7c7b0x30
7b0x2e770x7c790x30750x3f710x7c710x397f0x2c690x7c6e0x2e630x35740x3b3a0x28750x7
c780x2e7f0x3d710x7c6f0x2f3a0x296a
```

נראה שמדובר בפרוטוקול בסיסי שבו המשתמש מבקש קובץ ומקבל אותו מקודד. ה-XOR במהלך ההתקשרות מרמז שכנראה צריך להפעיל פעולת XOR באמצעות המפתח שמתקבל מהשרת על תוכן הקובץ כדי לקבל את ה-plaintext.

ננסה לחקות את הפרוטוקול בעצמנו (הקוד מצורף בשלמותו בעמוד הבא) ונקבל את התוצאה הבאה:

```
<< Welcome!
>> HELLO
<< HELLO
>> XOR
<< 0E18
>> /usr/7Op_sECreT.txt
<< 0x68740x6f7f0x75610x616d0x513b0x4e4e0x6b470x69280x5a470x476c0x2f65
Decoded Message:
bytearray(b'flag{you_#@Ve_g0T_It!}')
```

את הקוד אפשר לכתוב בצורה הרבה יותר קצרה, אבל זאת הזדמנות טובה לראות Context Manager בפעולה על מנת לשלוט בצורה נקייה בפתיחה ובסגירה של ה-Socket.

מחלקת Protocol מממשת את פרוטוקול התקשורת עם השרת (עם כמה הנחות בפונקציית recv). פונקציית decode_msg מפענחת את ההודעה על ידי מעבר על ההודעה בחלקים (כל חלק הוא שישה בתים - תחילית של 0x וארבעה בתים של מידע) וביצוע XOR עם המפתח שהתקבל בשלב הקודם.

```
import socket, re

class Protocol(object):
    def __init__(self, ip, port):
        self.ip = ip
        self.port = port
        self.msg_id = 0
        self.recv_reg =
           re.compile('^(?P<id>\d+) (?P<len>\d+) (?P<payload>.+)$')

    def __enter__(self):
        self.socket = socket.socket()
        self.socket.connect((self.ip, self.port))
```

```
        return self

    def __exit__(self, *args):
        self.socket.close()

    def log(self, msg):
        print(msg)

    def send(self, msg):
        self.log(">> {}".format(msg))
        self.msg_id += 1
        full_msg = "{} {} {}\n".format(self.msg_id, len(msg), msg)
        self.socket.send(full_msg.encode('UTF-8'))

    def recv(self):
        msg = self.socket.recv(1024)
        match = self.recv_reg.match(msg.decode('UTF-8'))
        if match:
            assert(int(match.group("id")) == self.msg_id)
            assert(int(match.group("len")) ==
len(match.group("payload")))
            self.log("<< {}".format(match.group("payload")))
            return match.group("payload")
        raise Exception("Unexpected format: {}".format(msg))

    def decode_msg(self, xor, msg):
        chunk_len = len("0x") + len(xor)
        frame = bytearray()
        for i in range(0, len(msg), chunk_len):
            s = msg[i:i + chunk_len]
            chunk_val = int(s, 16)
            after_xor = (chunk_val ^ int(xor, 16))
            for b in (after_xor.to_bytes(len(xor) // 2,
                                         byteorder='big',
                                         signed=True) ):
                frame.append(b)
        return frame

with Protocol('35.157.111.68', 20001) as p:
    msg = p.recv()
    assert(msg == "Welcome!")
    p.send("HELLO")
    msg = p.recv()
    assert(msg == "HELLO")
    p.send("XOR")
    xor_val = p.recv()
    p.send("/usr/7Op_sECreT.txt")
    encrypted_file = p.recv()
    print ("Decoded Message:")
    print (p.decode_msg(xor_val, encrypted_file))
```

אתגר 6: PNG++ (קטגוריית Logic, 30 נקודות)

הוראות האתגר:

This image was encrypted using a custom cipher. We managed to get most of its code here Unfortunately, while moving things around, someone spilled coffee all over key_transformator.py. Can you help us decrypt the image?

הקוד להצפנת התמונה הוא:

```
import key_transformator
import random
import string

key_length = 4

def generate_initial_key():
    return ''.join(random.choice(string.ascii_uppercase) for _ in
range(4))

def xor(s1, s2):
    res = [chr(0)]*key_length
    for i in range(len(res)):
        q = ord(s1[i])
        d = ord(s2[i])
        k = q ^ d
        res[i] = chr(k)
    res = ''.join(res)
    return res

def add_padding(img):
    l = key_length - len(img)%key_length
    img += chr(l)*l
    return img

with open('flag.png', 'rb') as f:
    img = f.read()

img = add_padding(img)
key = generate_initial_key()

enc_data = ''
for i in range(0, len(img), key_length):
    enc = xor(img[i:i+key_length], key)
    key = key_transformator.transform(key)
    enc_data += enc

with open('encrypted.png', 'wb') as f:
    f.write(enc_data)
```

כאשר אנחנו רואים שראשית מוגרל מפתח של ארבעה בתים, ולאחר מכן הקוד עובר על תוכן התמונה ומבצע XOR עם המפתח, מבצע מניפולציה על המפתח וחוזר על הפעולה.

התמונה עצמה נקראת encrypted.png וכאשר מנסים לפתוח אותה, מקבלים שגיאה שהקובץ אינו בפורמט המתאים. למזלנו, פורמט PNG מכיל Header ידוע מראש, שבאמצעותו ניתן לנחש מהו מפתח ההצפנה.

לפי האתר הזה:

```
A PNG file consists of a PNG signature followed by a series of chunks.

The first eight bytes of a PNG file always contain the following (decimal)
values:
   137 80 78 71 13 10 26 10
```

נבדוק את הקובץ שקיבלנו בעורך Hex:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	C7	1C	0B	13	42	47	5C	5F	50	4E	47	5B	18	07	0C	05
00000010	52	50	4B	58	53	51	4B	07	5C	54	4B	5A	55	A0	58	FC
00000020	B6	54	4D	5C	53	32	0F	10	19	56	4F	EF	D6	5C	AC	3E
00000030	5F	58	51	60	7B	3A	1A	33	11	5A	53	18	7B	5B	54	E3

על מנת לקבל את המפתח המקורי, נבצע XOR שוב מול הערך שאמור להיות שם לפי התקן:

```
>>> expected = "137 80 78 71 13 10 26 10"
>>> expected = "137 80 78 71 13 10 26 10".split(" ")
>>> current = "C7 1C 0B 13 42 47 5C 5F 50".split(" ")
>>> for (dec1, hex1) in zip(expected, current):
...     print(hex(int(dec1) ^ int(hex1, 16)) + " ", end='')
...
0x4e 0x4c 0x45 0x54 0x4f 0x4d 0x46 0x55 >>>
>>> print ("\x4e\x4c\x45\x54")
NLET
>>>
```

נראה שהמפתח הוא NLET (0x4e 0x4c 0x45 0x54). אנחנו רואים גם שב-Chunk הבא, המפתח הפך להיות "0x4f 0x4d 0x46 0x55", כלומר קידמנו כל ערך ב-1.

נבצע מספר שינויים קלים בקוד ההצפנה על מנת לבצע פענוח:

```python
# (Using original functions)

def transform(key):
    return "".join(map(lambda x: chr((ord(x)+1) % 256), key))

with open('encrypted.png', 'rb') as f:
    img = f.read()

key = "NLET"

dec_data = ''
for i in range(0, len(img), key_length):
    dec = xor(img[i:i+key_length], key)
    key = transform(key)
    dec_data += dec

with open('flag.png', 'wb') as f:
    f.write(dec_data)
```

והתוצאה:

אתגר 7: Test my Patience (קטגוריית Surprise, 50 נקודות)

הוראות האתגר:

> Hi there,
>
> We found This executable on the local watchmaker's computer.
>
> It is rumored that somehow the watchmaker was the only person who succeeded to crack it.
>
> Think you're as good as the watchmaker?
>
> **Note: This file is not malicious in any way**

קודם כל, תמיד מרגיע לראות הצהרה בסגנון "קובץ זה אינו נוזקה". נשמע אמין. זה זמן טוב להזכיר שבמסגרת אתגרים יוצא לא פעם להוריד קבצי הרצה, כלים, ספריות וכד' ומומלץ מאוד להפעיל הכל בתוך מכונה וירטואלית, על כל צרה שלא תבוא.

נריץ את הקובץ ונראה:

```
C:\Users\IEUser\Downloads>tmp.exe
Hi there! Welcome to the guessing game
Can you guess the number I'm thinking about?
Your guess> 1337
Wrong one..
Your guess>
```

מדובר במשחק ניחושים, התוכנה חושבת על מספר כלשהו ואנחנו צריכים לנחש מהו. אחרי מספר ניחושים (ארוכים, קצרים, שליליים, לא חוקיים וכד') אפשר לראות שלעיתים לוקח לתוכנה יחסית הרבה זמן להחזיר תשובה. יחד עם השם של האתגר, נראה שמדובר ב-Timing Attack.

הסבר קצר: כאשר התוכנה בודקת את הניחוש, היא משווה אותו מול המספר הנבחר. אם הספרה הראשונה של הניחוש שווה לספרה הראשונה של התשובה הנכונה, ההשוואה תקח קצת יותר זמן. כמובן שבאתגרים מהסוג הזה, לעיתים מוסיפים השהייה מלאכותית כל מנת להקל על המדידה.

נכתוב סקריפט שינסה את כל הספרות 0-9, יבדוק מתי התוצאה חזרה הכי לאט, וימשיך לספרה הבאה.

נריץ את הסקריפט (הקוד המלא בדף הבא) ונקבל:

```
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
[9.17199993133545, 9.17199993133545, 9.17199993133545, 9.157000064849854,
 9.171000003814697, 9.20300006866455, 9.187999963760376, 9.875, 9.1870000
36239624, 9.187000036239624]
WIP answer: 66452410709227
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Wrong one..
Your guess> Good job my friend!
[9.906999826431274, 9.890000104904175, 9.907000064849854, 9.8910000324249
27]
6645241070922 74
```

הקוד:

```
from subprocess import Popen, PIPE
import time

p = Popen(['tmp.exe'], stdout=PIPE, stderr=PIPE, stdin=PIPE, shell=True)
print p.stdout.readline()
print p.stdout.readline()

answer = ""

searching = True
while searching:
    time_arr = []
    for i in xrange(10):
        start = time.time()
        p.stdin.write(answer + str(i))
        p.stdin.write("\n")
        line = p.stdout.readline()
        end = time.time()
        print line.rstrip()
        if not "Wrong" in line:
            answer += str(i)
            searching = False
            break
        time_arr.append(end-start)
    print time_arr
    if searching:
        answer += str(time_arr.index(max(time_arr)))
        print "WIP answer: {}".format(answer)
print answer
```

אתגר 8: 0120343536 (קטגוריית Logic, 60 נקודות)

הוראות האתגר:

flag{IAAAA_$AYP_%CP_C_WIIX_BYWAOX}

Not so fast...

They say the only place where flags come before work is the dictionary, ours is no different

Note: flag letters are all capital

המילון מכיל רשימה של כמעט 40,000 מילים. ננסה להשתמש במילון על מנת לפצח את הדגל. ראשית נמיין את המילים במילון לפי אורך (אפשר להתעלם ממילים באורך גדול מ-6 כי אין כאלה בדגל):

```
msg = "IAAAA_$AYP_%CP_C_WIIX_BYWAOX"


d = defaultdict(list)
with open("dictionary.txt") as f:
    for line in f:
        line = line.rstrip()
        l = len(line)
        if l <= 6:
            d[l].append(line)
```

כעת נתחיל לחפש מילים במילון שמתאימות לתבנית של הדגל. המילה הראשונה שכדאי לתקוף היא IAAAA, מכיוון שנדיר למצוא מילים עם 4 אותיות זהות רצופות.

```
for w in d[5]:
    if (w[1] == w[2] == w[3] == w[4]):
        print (w)
# CEEEE, OHHHH
```

נהמר על OHHHH, כי נדיר לראות CEEEE בתחילת משפט.

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH ?H?? ??? ? ?OO? ???H??
```

המילה הבאה שכדאי לתקוף היא WIIX:

```
for w in d[4]:
    if (w[1] == w[2] and w[0] != w[3] and w[2] == 'O'):
        print (w)
# POOR
```

מצאנו רק מילה אחת שמתאימה:

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH ?H?? ??? ? POOR ??PH?R
```

נחפש את BYWAOX:

```
for w in d[6]:
    if (w[2] == 'P' and w[3] == 'H' and w[5] == 'R'):
        print (w)
# CIPHER
```

שוב, רק מילה אחת:

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH ?HI? ??? ? POOR CIPHER
```

כעת ל-$AYP:

```
for w in d[4]:
    if (w[1] == 'H' and w[2] == 'I'):
        print (w)
#CHIC, OHIO, THIS
```

נבחר ב-THIS בתור המילה שמסתדרת הכי טוב במשפט:

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH THIS ??S ? POOR CIPHER
```

אין הרבה מילים באורך 1:

```
for w in d[1]:
    if (w[0] != 'I'):
        print (w)
#A, C
```

נלך על -A (מה זה C?)

```
:IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH THIS ?AS A POOR CIPHER
```

ומי שלא ניחש עד עכשיו יכול לחפש את המילה האחרונה:

```
for w in d[3]:
    if (w[1] == 'A' and w[2] == 'S'):
        print (w)
#WAS
```

קיבלנו:

```
IAAAA_$AYP_%CP_C_WIIX_BYWAOX
OHHHH THIS WAS A POOR CIPHER
```

אתגר 9: Puzzle (קטגוריית Programming, 70 נקודות)

הוראות האתגר:

At last, we've found you! We must solve this puzzle, and according to the prophecy - you are the one to solve it.

This puzzle is weird. It consists of a board with 10 columns and 10 rows, so there are 100 pieces.Yet, each piece is weird! It has four 'slices' - a top slice, a right slice, a bottom slice and a left slice. Each slice consists of a number. For example, consider this piece:

```
------------
|   \ 12 /   |
| 5 \   / 3 |
|    /  \    |
|   / 4  \   |
------------
```

Its top is 12, its right is 3, its bottom is 4 and its left is 5. For the puzzle to be solved, all pieces must be sorted into the board, where each slice is equal to its adjacent slice. In addition, a slice that has no adjacent slice (that is, the slice is a part of the board's border), must be 0. Other slices are never 0. For example, the following board (with 4 pieces) is valid:

```
------------------------
|   \ 0  /  ||  \ 0  /  |
| 0 \   / 9 || 9 \   / 0 |
|    /  \   ||   /  \    |
|   / 17 \  ||  / 11 \   |
------------------------
------------------------
|   \ 17 /  ||  \ 11 /  |
| 0 \   / 6 || 6 \   / 0 |
|    /  \   ||   /  \    |
|   / 0  \  ||  / 0  \   |
------------------------
```

In the board above, all the border slices are equal to 0. Consider the top-left piece. Its right slice is equal to 9, and its adjacent slice (the left slice of the top-right piece) also equals 9.

Unfortunately, we have the pieces in a shuffled order. They are given in the following format:

cube_id, [slices]; cube_id, slices; ... cube_id, slices

Where cube_id is a number from 0 to 99, and slices include the numbers in the order: top, right, bottom, left. For instance, consider the following shuffled board:

```
-----------------------------------
|   \ 0  /  ||  \ 0  /  ||  \ 5  /  |
| 18\   / 12|| 19\   / 7 || 19\   / 0 |
|    /  \   ||   /  \   ||   /  \    |
|   / 2  \  ||  / 6  \  ||  / 0  \   |
-----------------------------------
-----------------------------------
|   \ 6  /  ||  \ 14 /  ||  \ 7  /  |
| 10\   / 2 || 10\   /  0|| 0 \   / 12|
|    /  \   ||   /  \   ||   /  \    |
|   / 9  \  ||  / 5  \  ||  / 0  \   |
-----------------------------------
```

```
-----------------------------------
|  \ 0  /  ||  \ 0  /  ||  \ 0  /  |
| 7 \  / 0 || 7 \  / 17|| 17\  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 18 \  ||  / 9  \  ||  / 14 \  |
-----------------------------------
```

A string describing the above board is the following one:

'0,[0, 12, 2, 18]; 1,[0, 7, 6, 19]; 2,[5, 0, 0, 19]; 3,[6, 2, 9, 10]; 4,[14, 0, 5, 10]; 5,[7, 12, 0, 0]; 6,[0, 0, 18, 7]; 7,[0, 17, 9, 7]; 8,[0, 0, 14, 17]'

We need you to solve the puzzle!

Provide us a string that looks exactly as follows:

cube_id, times_to_rotate_clockwise; cube_id, times_to_rotate_clockwise;... cube_id, times_to_rotate_clockwise

For example, a solution string will look like this:

2,2; 1,0; 6,0; 4,2; 3,0; 0,1; 8,2; 7,2; 5,3

The above string corresponds to the following (valid) puzzle:

```
-----------------------------------
|  \ 0  /  ||  \ 0  /  ||  \ 0  /  |
| 0 \  / 19|| 19\  / 7 || 7 \  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 5  \  ||  / 6  \  ||  / 18 \  |
-----------------------------------
-----------------------------------
|  \ 5  /  ||  \ 6  /  ||  \ 18 /  |
| 0 \  / 10|| 10\  / 2 || 2 \  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 14 \  ||  / 9  \  ||  / 12 \  |
-----------------------------------
-----------------------------------
|  \ 14 /  ||  \ 9  /  ||  \ 12 /  |
| 0 \  / 17|| 17\  / 7 || 7 \  / 0 |
|   /  \   ||   /  \   ||   /  \   |
|  / 0  \  ||  / 0  \  ||  / 0  \  |
-----------------------------------
```

Consider the top-left piece. In the string, it corresponds to '2,2', as we take cube number 2 from the input:

2,[5, 0, 0, 19]

But we rotate it clock-wise, twice, so we get [0,19,5,0].

Now consider the top-middle piece. In the string, it corresponds to '1,0'. That is, we take cube number 1 from the input:

1,[0, 7, 6, 19]

And we don't rotate it at all (that is, rotate it 0 times) - as it's already in the right direction.

Got it?

Help us solve the puzzle!

The puzzle we have is:

0,[3, 19, 5, 15]; 1,[0, 17, 6, 11]; 2,[12, 15, 9, 5]; 3,[10, 2, 0, 7]; 4,[6, 8, 4, 0]; 5,[3, 1, 12, 17]; 6,[20, 16, 0, 0]; 7,[0, 1, 9, 0]; 8,[17, 16, 0, 8]; 9,[18, 15, 15, 17]; 10,[4, 9, 8, 16]; 11,[0, 11, 17, 20]; 12,[5, 6, 5, 19]; 13,[10, 11, 1, 4]; 14,[16, 2, 3, 5]; 15,[9, 20, 10, 11]; 16,[11, 3, 13, 3]; 17,[0, 2, 16, 2]; 18,[11, 18, 16, 5]; 19,[11, 20, 13, 15]; 20,[16, 18, 11, 1]; 21,[10, 8, 12, 3]; 22,[17, 18, 17, 18]; 23,[7, 17, 0, 17]; 24,[20, 16, 18, 4]; 25,[2, 14, 4, 13]; 26,[1, 6, 7, 2]; 27,[18, 8, 6, 9]; 28,[6, 10, 12, 16]; 29,[2, 20, 11, 20]; 30,[1, 5, 12, 10]; 31,[2, 7, 10, 9]; 32,[8, 17, 11, 12]; 33,[0, 11, 12, 20]; 34,[15, 2, 0, 3]; 35,[18, 10, 10, 8]; 36,[14, 6, 17, 9]; 37,[15, 7, 3, 8]; 38,[15, 3, 6, 0]; 39,[4, 11, 2, 15]; 40,[0, 5, 1, 1]; 41,[14, 10, 15, 8]; 42,[3, 8, 18, 5]; 43,[8, 11, 0, 13]; 44,[3, 11, 13, 8]; 45,[17, 1, 4, 2]; 46,[2, 13, 2, 0]; 47,[20, 0, 16, 18]; 48,[8, 13, 15, 17]; 49,[4, 13, 8, 8]; 50,[19, 20, 17, 5]; 51,[5, 19, 8, 1]; 52,[13, 17, 4, 5]; 53,[15, 0, 16, 8]; 54,[5, 4, 1, 2]; 55,[7, 11, 0, 15]; 56,[9, 12, 4, 7]; 57,[12, 7, 8, 8]; 58,[2, 17, 12, 19]; 59,[1, 9, 3, 6]; 60,[12, 10, 8, 19]; 61,[4, 11, 11, 5]; 62,[0, 17, 17, 13]; 63,[0, 4, 12, 8]; 64,[16, 20, 11, 4]; 65,[0, 18, 20, 15]; 66,[9, 6, 11, 8]; 67,[4, 5, 15, 18]; 68,[8, 7, 19, 11]; 69,[20, 11, 5, 0]; 70,[3, 0, 2, 8]; 71,[13, 11, 0, 2]; 72,[0, 13, 5, 17]; 73,[13, 5, 0, 2]; 74,[2, 0, 17, 7]; 75,[7, 9, 16, 7]; 76,[11, 16, 8, 1]; 77,[18, 19, 12, 6]; 78,[2, 7, 20, 2]; 79,[9, 15, 19, 8]; 80,[0, 11, 12, 15]; 81,[8, 20, 4, 18]; 82,[17, 0, 20, 13]; 83,[7, 18, 0, 4]; 84,[11, 10, 8, 8]; 85,[15, 17, 1, 15]; 86,[9, 8, 7, 12]; 87,[1, 13, 11, 3]; 88,[3, 19, 11, 6]; 89,[20, 17, 0, 16]; 90,[5, 12, 17, 2]; 91,[12, 16, 0, 15]; 92,[18, 12, 8, 2]; 93,[13, 0, 0, 11]; 94,[18, 8, 4, 1]; 95,[7, 0, 5, 4]; 96,[3, 11, 20, 14]; 97,[2, 10, 18, 10]; 98,[11, 4, 0, 9]; 99,[0, 0, 17, 17]

Good luck!

הפתרון לתרגיל הזה הוא קצת Overkill, כולל לוגיקה לציור החלקים, פשוט כי זה היה תרגיל תכנותי נחמד.

הגישה העקרונית היא פתרון באמצעות Backtracking - כמו שבדרך כלל פותרים מבוך, או את בעיית 8 המלכות. בכל צעד, ננסה להציב חלק מתאים אחד על הלוח. אם נגלה שאין חלקים מתאימים עבור הצעד הנוכחי - נחזור אחורה, נבטל את ההצבה, וננסה להתקדם עם חלק מתאים אחר. כמו בכל רקורסיה, לפעמים זה מרגיש קצת כמו קסם.

ראשית, נייצר ייצוג לחלק בודד מהפאזל:

```
UP    = 0
RIGHT = 1
DOWN  = 2
LEFT  = 3

class Piece(object):
    def __init__(self, id, slices):
        self.used = False
        self.id = id
        self.slices = collections.deque(slices)

        self.rep_str = "_{}_{}_{}_{}_{}_".format(self.left, self.up, self.right,
self.down, self.left)

        self.rotations = 0

        self.is_border = False
        self.is_corner = False
        num_zeroes = self.slices.count(0)
        if num_zeroes == 1:
            self.is_border = True
        elif num_zeroes == 2:
            self.is_corner = True

    def rotate(self):
        self.rotations += 1
        self.slices.rotate(1)
```

```
    def rotate_until_1(self, direction1, value1):
        while self.slices[direction1] != value1:
            self.rotate()

    def rotate_until_2(self, direction1, value1, direction2, value2):
        while self.slices[direction1] != value1 or self.slices[direction2] !=
value2:
            self.rotate()

    @property
    def up(self):
        return self.slices[UP]

    @property
    def right(self):
        return self.slices[RIGHT]

    @property
    def down(self):
        return self.slices[DOWN]

    @property
    def left(self):
        return self.slices[LEFT]

    def __repr__(self):
        return "Piece({}, [{}])".format(self.id, list(self.slices))

    def __str__(self):
        ret =  "------------\n"
        ret += "|  \\ {:02} /  |\n".format(self.up)
        ret += "| {:02}\\  / {:02}|\n".format(self.left, self.right)
        ret += "|   /  \\   |\n"
        ret += "|  / {:02} \\  |\n".format(self.down)
        ret +=  "------------"
        return ret

    def __hash__(self):
        return hash(self.id)

    def __eq__(self, other):
        if isinstance(self, other.__class__):
            return self.id == other.id
        return False
```

רוב המחלקה הזו סטנדרטית לחלוטין, אבל יש מספר נקודות שכדאי להתעכב עליהן:

1. קיימות שלוש מתודות לסיבוב חלקים:
   a. **rotate** - מסובבת את החלק פעם אחת
   b. **rotate_until_1** - מסובבת חלק עד שכיוון מסויים מקבל ערך נתון
   c. **rotate_until_2** - מסובבת חלק עד ששני כיוונים מקבלים שני ערכים נתונים. למשל, בהנתן החלק [0, 12, 2, 18], ניתן לסובב אותו עד שה-12 יופיע למעלה באמצעות מתודת rotate_until_1.
2. כל חלק יודע אם הוא פינה, גבול או חלק פנימי לפי מספר האפסים.
3. קיים ייצוג אלטרנטיבי לכל חלק בדמות:

```
self.rep_str = "_{}_{}_{}_{}_{}_".format(self.left, self.up, self.right,
self.down, self.left)
```

הייצוג הזה מועיל לחיפוש חלקים שיש להם שני מספרים סמוכים. למשל, כדי לבדוק אם החלק [18, 2, 12, 0] כולל 18 ליד 0, אפשר לחפש "_18_0_" בייצוג "_18_0_12_2_18_".

החלק הבא הוא ייצוג של לוח, וגם הוא יחסית סטנדרטי:

```
BLANK_PIECE = Piece(-1, [-1, -1, -1, -1])

class Board(object):
    def __init__(self, rows, cols):
        self.rows = rows
        self.cols = cols

        self.pieces   = [[BLANK_PIECE for x in range(self.cols)] for y in
range(self.rows)]
        self.corners = set()
        self.borders = set()
        self.inner   = set()


    def Place_piece(self, row, col, new_piece):
        self.pieces[row][col] = new_piece

        if new_piece.is_corner:
            self.corners.add(new_piece)
        elif new_piece.is_border:
            self.borders.add(new_piece)
        else:
            self.inner.add(new_piece)

    def Print(self):
        for i in range(self.rows):
            for j in range(self.cols):
                print (str(self.pieces[i][j]))

    def Print_corners(self):
        for piece in self.corners:
            print (str(piece))

    def __str__(self):
        ret = ""
        for i in range(self.rows):
            ret += ("------------" * self.cols) + "\n"
            for j in range(self.cols):
                ret += "|  \\ {:02} /  |".format(self.pieces[i][j].up)
            ret += "\n"
            for j in range(self.cols):
                 ret += "| {:02}\\  / {:02}|".format(self.pieces[i][j].left,
self.pieces[i][j].right)
            ret += "\n"
            ret += ("|   /  \\   |" * self.cols) + "\n"
            for j in range(self.cols):
                ret += "|  / {:02} \\  |".format(self.pieces[i][j].down)
            ret += "\n"
            ret += ("------------" * self.cols) + "\n"
        return ret
```

אפשר להציב חלק, להדפיס את הלוח וזהו פחות או יותר. כדאי לציין שהלוח ממיין את החלקים שבו לפינות, גבולות וחלקים פנימיים, לצורך גישה נוחה יותר בזמן ריצה.

עוד פונקציית עזר מנסה למצוא את החלק המתאים ביותר לשמש בתור הפינה הראשונה שתוצב על הלוח (האלגוריתם בנוי כך שהוא מתחיל להציב חלקים מהפינה השמאלית העליונה):

```
def find_best_corner(board):
    for corner in board.corners:
        corner.rotate_until_2(LEFT, 0, UP, 0)
    for border in board.borders:
        border.rotate_until_1(UP, 0)

    candidates = collections.defaultdict(int)
    for corner in board.corners:
        for border in board.borders:
            if border.left == corner.right:
                candidates[corner] += 1
    #print (candidates)
    return min(candidates, key=candidates.get)
```

הפונקציה מחפשת את הפינה שיש עבורה הכי מעט מועמדים לגבול שמתאימים להצבה ליד הפינה. השלב הזה לא הכרחי (אפשר פשוט לנסות את כל הפינות) אבל עשוי לעזור קצת.

המחלקה הבאה משמשת למציאת הפתרון על ידי Backtracking:

```
class BoardManager(object):
    def __init__(self, board, solution):
        self.board = board
        self.solution = solution
        self.num_pieces = self.board.rows * self.board.cols
        self.num_placed_pieces = 0

        self.pool = [self.board.inner, self.board.borders, self.board.corners]

    def place_sol(self, row, col, piece):
        piece.used = True
        self.solution.Place_piece(row, col, piece)
        if piece.is_corner:
            self.board.corners.remove(piece)
        elif piece.is_border:
            self.board.borders.remove(piece)
        else:
            self.board.inner.remove(piece)

    def remove_sol(self, row, col):
        piece = self.solution.pieces[row][col]
        piece.used = False
        self.solution.Place_piece(row, col, BLANK_PIECE)
        if piece.is_corner:
            self.board.corners.add(piece)
        elif piece.is_border:
            self.board.borders.add(piece)
        else:
            self.board.inner.add(piece)

    def get_candidates(self, row, col):
        expected_up = 0 if row == 0 else self.solution.pieces[row-1][col].down
        expected_left = 0 if col == 0 else self.solution.pieces[row][col-
1].right
        expected_right = 0 if col == self.solution.cols - 1 else -1
        expected_down = 0 if row == self.solution.rows - 1 else -1

        pool = self.pool[[expected_up, expected_left, expected_right,
expected_down].count(0)]
        filter_str = "_{}_{}_".format(expected_left, expected_up)
        candidates = list(filter(lambda x: filter_str in x.rep_str, pool))
```

```
        return (candidates, expected_left, expected_up)


    def get_next_coord(self, row, col):
        col += 1
        if col == self.solution.cols:
            row += 1
            col = 0
        if row == self.solution.rows:
            return (-1, -1)
        else:
            return (row, col)

    def place_one_peice(self, row, col):
        if row == -1 and col == -1:
            print(str(self.solution))
            return True


        (candidates, expected_left, expected_up) = self.get_candidates(row, col)
        if len(candidates) == 0:
            return False

        res = False
        for candidate in candidates:
            candidate.rotate_until_2(LEFT, expected_left, UP, expected_up)
            self.place_sol(row, col, candidate)
            res = self.place_one_peice(*self.get_next_coord(row, col))
            if res:
                return True
            self.remove_sol(row, col)

        return False
```

מתודת place_sol מוציאה חלק מהמאגר ומניחה אותו על לוח הפתרון. מתודת remove_sol מסירה חלק מלוח הפתרון ומחזירה אותו למאגר החלקים.

מתודת get_candidates מקבלת מיקום על הלוח, ובודקת אילו מועמדים המתאימים למיקום זה קיימים במאגר החלקים. המתודה תחזיר רק חלקים שאפשר לסדר אותם כך שהמספר העליון שלהם יתאים למספר התחתון של החלק מעליהם, והמספר השמאלי שלהם יתאים למספר הימני של החלק משמאל.

מתודת get_next_coord היא מתודת עזר למעבר על הלוח - היא מקבל מיקום ומחזירה את המיקום הבא שבו יש להציב חלק (מכיוון שהלוח הוא דו-מימדי, נוח שתהיה מתודת עזר למעבר בין שורות).

לבסוף, מתודת place_one_piece היא המתודה הרקורסיבית שמבצעת את ה-Backtracking: היא מנסה להציב חלק מתאים אחד על הלוח (באמצעות המועמדים שהיא קיבלה מ-get_candidates) ואז ממשיכה אל המיקום הבא בלוח. אם היא מקבלת תשובה (מקריאה רקורסיבית של עצמה) שהניסיון נכשל, היא מסירה את החלק שהיא הציבה כעת ומנסה מועמד אחר. תנאי העצירה הוא אם החלק הבא שיש להציב נמצא מחוץ ללוח. במקרה כזה, "מדווחים אחורה" שההצבה הצליחה וכל הקריאות "מתקפלות".

על מנת להתניע את התהליך, יש צורך בקוד הבא:

```
if __name__ == "__main__":
    with open('puzzle.txt','r') as f:
        input = f.read()
        total_pieces = input.count(";") + 1
        rows = int(math.sqrt(total_pieces))
        cols = rows
        print ("Matrix of {} rows, {} cols".format(rows, cols))
        b = Board(rows, cols)
        match_iter = re.finditer(r"(\d+),\[(\d+), (\d+), (\d+), (\d+)\];?\s*",
input)
        for i in range(rows):
            for j in range(cols):
                new_input = next(match_iter)
                new_piece = Piece(int(new_input.group(1)),
                                  [int(x) for x in new_input.groups()[1:]])
                b.Place_piece(i, j, new_piece)

        print(str(b))

        print ("-=-=-=-=-=" * 3)

        s = Board(rows, cols)

        bm = BoardManager(b, s)


        # Top Left corner
        first_corner = find_best_corner(b)
        first_corner.rotate_until_2(LEFT, 0, UP, 0)
        bm.place_sol(0, 0, first_corner)

        sol_arr = []
        if (bm.place_one_peice(0, 1)):
            # Found solution, build representation:
            for row in range(bm.solution.rows):
                for col in range(bm.solution.cols):
                    piece = bm.solution.pieces[row][col]
                    sol_arr.append("{},{}".format(piece.id, piece.rotations %
4))

        print ("; ".join(sol_arr))
```

הקוד בונה את הלוחות, מוצא מועמד מוביל לפינה השמאלית-עליונה ואז קורא למתודה הרקורסיבית להמשך התהליך. במידה ונמצאה תוצאה, הקוד בונה את הייצוג המתאים ומדפיס אותו למסך.

הלוח המקורי:

```
Matrix of 10 rows, 10 cols
-----------------------------------------------------------------------------------------------------------------
|  \ 03 /  ||  \ 00 /  ||  \ 12 /  ||  \ 10 /  ||  \ 06 /  ||  \ 03 /  ||  \ 20 /  ||  \ 00 /  ||  \ 17 /  ||  \ 18 /  |
| 15\  / 19|| 11\  / 17|| 05\  / 15|| 07\  / 02|| 00\  / 08|| 17\  / 01|| 00\  / 16|| 00\  / 01|| 08\  / 16|| 17\  / 15|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 05 \  ||  / 06 \  ||  / 09 \  ||  / 00 \  ||  / 04 \  ||  / 12 \  ||  / 00 \  ||  / 09 \  ||  / 00 \  ||  / 15 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 04 /  ||  \ 00 /  ||  \ 05 /  ||  \ 10 /  ||  \ 16 /  ||  \ 09 /  ||  \ 11 /  ||  \ 00 /  ||  \ 11 /  ||  \ 11 /  |
| 16\  / 09|| 20\  / 11|| 19\  / 06|| 04\  / 11|| 05\  / 02|| 11\  / 20|| 03\  / 03|| 02\  / 02|| 05\  / 18|| 15\  / 20|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 08 \  ||  / 17 \  ||  / 05 \  ||  / 01 \  ||  / 03 \  ||  / 10 \  ||  / 13 \  ||  / 16 \  ||  / 16 \  ||  / 13 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 16 /  ||  \ 10 /  ||  \ 17 /  ||  \ 07 /  ||  \ 20 /  ||  \ 02 /  ||  \ 01 /  ||  \ 18 /  ||  \ 06 /  ||  \ 02 /  |
| 01\  / 18|| 03\  / 08|| 18\  / 18|| 17\  / 17|| 04\  / 16|| 13\  / 14|| 02\  / 06|| 09\  / 08|| 16\  / 10|| 20\  / 20|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 11 \  ||  / 12 \  ||  / 17 \  ||  / 00 \  ||  / 18 \  ||  / 04 \  ||  / 07 \  ||  / 06 \  ||  / 12 \  ||  / 11 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 01 /  ||  \ 02 /  ||  \ 08 /  ||  \ 00 /  ||  \ 15 /  ||  \ 18 /  ||  \ 14 /  ||  \ 15 /  ||  \ 15 /  ||  \ 04 /  |
| 10\  / 05|| 09\  / 07|| 12\  / 17|| 20\  / 11|| 03\  / 02|| 08\  / 10|| 09\  / 06|| 08\  / 07|| 00\  / 03|| 15\  / 11|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 12 \  ||  / 10 \  ||  / 11 \  ||  / 12 \  ||  / 00 \  ||  / 10 \  ||  / 17 \  ||  / 03 \  ||  / 06 \  ||  / 02 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 00 /  ||  \ 14 /  ||  \ 03 /  ||  \ 08 /  ||  \ 03 /  ||  \ 17 /  ||  \ 02 /  ||  \ 20 /  ||  \ 08 /  ||  \ 04 /  |
| 01\  / 05|| 08\  / 10|| 05\  / 08|| 13\  / 11|| 08\  / 11|| 02\  / 01|| 00\  / 13|| 18\  / 00|| 17\  / 13|| 08\  / 13|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 01 \  ||  / 15 \  ||  / 18 \  ||  / 00 \  ||  / 13 \  ||  / 04 \  ||  / 02 \  ||  / 16 \  ||  / 15 \  ||  / 08 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 19 /  ||  \ 05 /  ||  \ 13 /  ||  \ 15 /  ||  \ 05 /  ||  \ 07 /  ||  \ 09 /  ||  \ 12 /  ||  \ 02 /  ||  \ 01 /  |
| 05\  / 20|| 01\  / 19|| 05\  / 17|| 08\  / 00|| 02\  / 04|| 15\  / 11|| 07\  / 12|| 08\  / 07|| 19\  / 17|| 06\  / 09|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 17 \  ||  / 08 \  ||  / 04 \  ||  / 16 \  ||  / 01 \  ||  / 00 \  ||  / 04 \  ||  / 08 \  ||  / 12 \  ||  / 03 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 12 /  ||  \ 04 /  ||  \ 00 /  ||  \ 00 /  ||  \ 16 /  ||  \ 00 /  ||  \ 09 /  ||  \ 04 /  ||  \ 08 /  ||  \ 20 /  |
| 19\  / 10|| 05\  / 11|| 13\  / 17|| 08\  / 04|| 04\  / 20|| 15\  / 18|| 08\  / 06|| 18\  / 05|| 11\  / 07|| 00\  / 11|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 08 \  ||  / 11 \  ||  / 17 \  ||  / 12 \  ||  / 11 \  ||  / 20 \  ||  / 11 \  ||  / 15 \  ||  / 19 \  ||  / 05 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 03 /  ||  \ 13 /  ||  \ 00 /  ||  \ 13 /  ||  \ 02 /  ||  \ 07 /  ||  \ 11 /  ||  \ 18 /  ||  \ 02 /  ||  \ 09 /  |
| 08\  / 00|| 02\  / 11|| 17\  / 13|| 02\  / 05|| 07\  / 00|| 07\  / 09|| 01\  / 16|| 06\  / 19|| 02\  / 07|| 08\  / 15|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 02 \  ||  / 00 \  ||  / 05 \  ||  / 00 \  ||  / 17 \  ||  / 16 \  ||  / 08 \  ||  / 12 \  ||  / 20 \  ||  / 19 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 00 /  ||  \ 08 /  ||  \ 17 /  ||  \ 07 /  ||  \ 11 /  ||  \ 15 /  ||  \ 09 /  ||  \ 01 /  ||  \ 03 /  ||  \ 20 /  |
| 15\  / 11|| 18\  / 20|| 13\  / 00|| 04\  / 18|| 08\  / 10|| 15\  / 17|| 12\  / 08|| 03\  / 13|| 06\  / 19|| 16\  / 17|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 12 \  ||  / 04 \  ||  / 20 \  ||  / 00 \  ||  / 08 \  ||  / 01 \  ||  / 07 \  ||  / 11 \  ||  / 11 \  ||  / 00 \  |
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
|  \ 05 /  ||  \ 12 /  ||  \ 18 /  ||  \ 13 /  ||  \ 18 /  ||  \ 07 /  ||  \ 03 /  ||  \ 02 /  ||  \ 11 /  ||  \ 00 /  |
| 02\  / 12|| 15\  / 16|| 02\  / 12|| 11\  / 00|| 01\  / 08|| 04\  / 00|| 14\  / 11|| 10\  / 10|| 09\  / 04|| 17\  / 00|
|   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    ||   / \    |
|  / 17 \  ||  / 00 \  ||  / 08 \  ||  / 00 \  ||  / 04 \  ||  / 05 \  ||  / 20 \  ||  / 18 \  ||  / 00 \  ||  / 17 \  |
-----------------------------------------------------------------------------------------------------------------
```

הפתרון:

```
-----------------------------------------------------------------------------------------------------------------------
|  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  ||  \ 00 /  |
| 00\  / 01|| 01\  / 05|| 05\  / 02|| 02\  / 07|| 07\  / 05|| 05\  / 20|| 20\  / 11|| 11\  / 15|| 15\  / 16|| 16\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 09 \  ||  / 01 \  ||  / 13 \  ||  / 10 \  ||  / 04 \  ||  / 11 \  ||  / 12 \  ||  / 07 \  ||  / 08 \  ||  / 20 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 09 /  ||  \ 01 /  ||  \ 13 /  ||  \ 10 /  ||  \ 04 /  ||  \ 11 /  ||  \ 12 /  ||  \ 07 /  ||  \ 08 /  ||  \ 20 /  |
| 00\  / 11|| 11\  / 04|| 04\  / 02|| 02\  / 18|| 18\  / 05|| 05\  / 18|| 18\  / 08|| 08\  / 12|| 12\  / 17|| 17\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 04 \  ||  / 10 \  ||  / 14 \  ||  / 10 \  ||  / 15 \  ||  / 16 \  ||  / 02 \  ||  / 09 \  ||  / 11 \  ||  / 11 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 04 /  ||  \ 10 /  ||  \ 14 /  ||  \ 10 /  ||  \ 15 /  ||  \ 16 /  ||  \ 02 /  ||  \ 09 /  ||  \ 11 /  ||  \ 11 /  |
| 00\  / 07|| 07\  / 09|| 09\  / 06|| 06\  / 12|| 12\  / 09|| 09\  / 07|| 07\  / 01|| 01\  / 03|| 03\  / 13|| 13\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 18 \  ||  / 02 \  ||  / 17 \  ||  / 16 \  ||  / 05 \  ||  / 07 \  ||  / 06 \  ||  / 06 \  ||  / 08 \  ||  / 02 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 18 /  ||  \ 02 /  ||  \ 17 /  ||  \ 16 /  ||  \ 05 /  ||  \ 07 /  ||  \ 06 /  ||  \ 06 /  ||  \ 08 /  ||  \ 02 /  |
| 00\  / 20|| 20\  / 20|| 20\  / 05|| 05\  / 02|| 02\  / 04|| 04\  / 09|| 09\  / 11|| 11\  / 03|| 03\  / 15|| 15\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 15 \  ||  / 11 \  ||  / 19 \  ||  / 03 \  ||  / 01 \  ||  / 12 \  ||  / 08 \  ||  / 19 \  ||  / 07 \  ||  / 03 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 15 /  ||  \ 11 /  ||  \ 19 /  ||  \ 03 /  ||  \ 01 /  ||  \ 12 /  ||  \ 08 /  ||  \ 19 /  ||  \ 07 /  ||  \ 03 /  |
| 00\  / 03|| 03\  / 03|| 03\  / 05|| 05\  / 08|| 08\  / 05|| 05\  / 10|| 10\  / 18|| 18\  / 12|| 12\  / 08|| 08\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 06 \  ||  / 13 \  ||  / 15 \  ||  / 18 \  ||  / 19 \  ||  / 01 \  ||  / 10 \  ||  / 06 \  ||  / 08 \  ||  / 02 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 06 /  ||  \ 13 /  ||  \ 15 /  ||  \ 18 /  ||  \ 19 /  ||  \ 01 /  ||  \ 10 /  ||  \ 06 /  ||  \ 08 /  ||  \ 02 /  |
| 00\  / 08|| 08\  / 15|| 15\  / 17|| 17\  / 15|| 15\  / 08|| 08\  / 11|| 11\  / 08|| 08\  / 09|| 09\  / 16|| 16\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 04 \  ||  / 17 \  ||  / 01 \  ||  / 15 \  ||  / 09 \  ||  / 16 \  ||  / 08 \  ||  / 18 \  ||  / 04 \  ||  / 02 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 04 /  ||  \ 17 /  ||  \ 01 /  ||  \ 15 /  ||  \ 09 /  ||  \ 16 /  ||  \ 08 /  ||  \ 18 /  ||  \ 04 /  ||  \ 02 /  |
| 00\  / 12|| 12\  / 03|| 03\  / 13|| 13\  / 11|| 11\  / 20|| 20\  / 18|| 18\  / 04|| 04\  / 08|| 08\  / 13|| 13\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 08 \  ||  / 01 \  ||  / 11 \  ||  / 20 \  ||  / 10 \  ||  / 04 \  ||  / 01 \  ||  / 20 \  ||  / 08 \  ||  / 02 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 08 /  ||  \ 01 /  ||  \ 11 /  ||  \ 20 /  ||  \ 10 /  ||  \ 04 /  ||  \ 01 /  ||  \ 20 /  ||  \ 08 /  ||  \ 02 /  |
| 00\  / 17|| 17\  / 04|| 04\  / 11|| 11\  / 14|| 14\  / 15|| 15\  / 11|| 11\  / 16|| 16\  / 11|| 11\  / 07|| 07\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 16 \  ||  / 02 \  ||  / 05 \  ||  / 03 \  ||  / 08 \  ||  / 02 \  ||  / 18 \  ||  / 04 \  ||  / 19 \  ||  / 17 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 16 /  ||  \ 02 /  ||  \ 05 /  ||  \ 03 /  ||  \ 08 /  ||  \ 02 /  ||  \ 18 /  ||  \ 04 /  ||  \ 19 /  ||  \ 17 /  |
| 00\  / 20|| 20\  / 02|| 02\  / 12|| 12\  / 10|| 10\  / 19|| 19\  / 17|| 17\  / 17|| 17\  / 05|| 05\  / 05|| 05\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 17 \  ||  / 07 \  ||  / 17 \  ||  / 08 \  ||  / 12 \  ||  / 12 \  ||  / 18 \  ||  / 13 \  ||  / 06 \  ||  / 13 \  |
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
|  \ 17 /  ||  \ 07 /  ||  \ 17 /  ||  \ 08 /  ||  \ 12 /  ||  \ 12 /  ||  \ 18 /  ||  \ 13 /  ||  \ 06 /  ||  \ 13 /  |
| 00\  / 17|| 17\  / 17|| 17\  / 13|| 13\  / 11|| 11\  / 15|| 15\  / 16|| 16\  / 20|| 20\  / 17|| 17\  / 11|| 11\  / 00|
|   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   ||   / \   |
|  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  ||  / 00 \  |
-----------------------------------------------------------------------------------------------------------------------
```

הייצוג:

```
7,0; 40,0; 73,2; 3,2; 95,3; 69,1; 33,0; 55,2; 53,3; 6,2; 98,1; 13,2; 25,1; 97,3;
67,0; 18,0; 92,3; 86,2; 32,0; 11,1; 83,1; 31,2; 36,0; 28,3; 2,3; 75,2; 26,1;
59,3; 44,3; 71,3; 65,3; 29,0; 50,2; 14,0; 54,0; 56,1; 66,3; 88,1; 37,1; 34,3;
38,0; 16,0; 0,3; 42,0; 51,1; 30,2; 35,1; 77,3; 57,3; 70,0; 4,0; 48,3; 85,0; 9,0;
79,2; 76,1; 84,3; 27,2; 10,2; 17,1; 63,3; 5,1; 87,0; 19,1; 15,0; 24,3; 94,3;
81,1; 49,0; 46,2; 8,1; 45,3; 61,3; 96,2; 41,3; 39,0; 20,1; 64,3; 68,0; 74,0;
89,1; 78,1; 90,0; 21,1; 60,2; 58,0; 22,1; 52,2; 12,1; 72,1; 99,2; 23,0; 62,2;
43,0; 80,2; 91,0; 47,1; 82,1; 1,2; 93,0
```

אתגר 10: Simple Machine 2 (קטגוריית Reversing, 85 נקודות)

תיאור האתגר:

A Simple Machine
What is this?
You stand before assembly code for a custom Virtual Machine. You will find the flag once you understand the code. Everything you need to know is described below. Don't forget to check ou the example code!
Get the machine code Here

Top level description
The machine is stack based, which means most operations pop data off the top of the stack and push the result. for further reference, https://en.wikipedia.org/wiki/Stack_machine#Advantages_of_stack_machine_instruction_sets
The machine state is defined by an Instruction Pointer, and a Stack data structure. The next instruction to be executed is pointed to by IP, and it generally reads/write values from/to the top of the stack. Every opcode is exactly 1 byte in size. The program is read and executed sequentially starting at offset 0 in the file. Execution stops if an invalid stack address is referenced or the IP is out of code bounds.

Instruction Set
Important!
IP is incremented as the instruction is read (before decode/execute). This increment is not mentioned in the instruction pseudo-code. Therefore, every instruction that adds an offset to IP will result in IP = IP + offset + 1.
An instruction that resets IP as IP = new_value discards the increment.

Instruction Pseudo Code Notations
stack.push([value]) - pushes the value to the stack
stack.pop() - dequeue the last value pushed to the stack .
a = stack.pop() - dequeue the last value pushed to the stack, save value to pseudo-variable 'a'.
stack.empty() - true if there are no more values on the stack, false otherwise
stack[N] - the value of the Nth element on the stack
IP - the instruction pointer.

Stack Instructions:
Push <value>
- opcode is 0x80 + value
- Pushes the value to the stack, stack[0] is now , stack[1] is now the previous stack[0] value, and so on.
- value <= 0x7f
- Push 0x32 is encoded as 0xB2.

stack.push(value)

Load <offset>
- opcode is 0x40 + offset
- Pushes the value at stack[offset] to the stack.
- value <= 0x3f
- Load 0x12 is encoded as 0x52.
- Loading from an offset out of bounds (i.e pushing 10 values and loading from offset 12) will cause a fault and execution will terminate.

stack.push(stack[offset])

Pop
- opcode 0x20
- Same encoding as Swap 0
- Swap 0 is an empty statement, thus this opcode pops a value from the stack without doing anything with it.

stack.pop()

Swap <index>
- opcode is 0x20 + index
- Swaps the element at HEAD with the element at index.
- 1 <= index < 0x20.

- Swap 3 is encoded as 0x23.

temp = stack[index]
stack[index] = stack.pop()
stack.push(temp)

Arithmetic instructions
These instructions read 2 values off the stack and push the result. ### Single outupt instructions:
Add

- opcode is 0x00.
- operands are viewed as signed bytes

stack.push(stack.pop() + stack.pop())

Subtract

- opcode is 0x01.
- operands are viewed as signed bytes

stack.push(stack.pop() - stack.pop())

Multiply

- opcode is 0x03.
- operands are viewed as signed bytes

stack.push(stack.pop() * stack.pop())

2-byte output
Divide

- opcode is 0x02.
- division reminder is at HEAD, division result follows
- operands are viewed as unsigned bytes

a = stack.pop()
b = stack.pop()
stack.push(a / b)
stack.push(a % b)

Flow Control Instructions:
These instructions change the Instruction Pointer and allow for loops, function calls, etc.
Jump

- opcode is 0x10. Jumps to offset stack[0].
- offset is signed! Jumping to a negative offset is a jump backwards.
- Pops an offset from the stack, adds it to IP.

IP = IP + stack.pop()

Call

- opcode is 0x11. Jumps to stack[0], saves origin.
- same as Jump, only IP before execution is pushed.
- offset is signed! Calling to a negative offset is a jump backwards.

offset = stack.pop()
stack.push(IP) ; note that IP was already incremented here, points to next instruction.
IP = IP + offset

Ret

- opcode is 0x12. Pops value from the stack, moves IP to the popped value.

IP = stack.pop()

CJE

- opcode is 0x14. Jumps to stack[0] if stack[1] == stack[2]. pops all values either way.
- offset is signed! Jumping to a negative offset is a jump backwards.

offset = stack.pop()
if stack.pop() == stack.pop():
  IP = IP + offset

JSE

- opcode is 0x18. Adds stack[0] to IP if it is the last value on the stack.

```
offset = stack.pop()
if stack.empty():
   IP = IP + offset
```

Input Output Instructions:
These instructions either output an ASCII byte or read an ASCII byte from the input/output device.
Read

- opcode is 0x08
- Waits for a single byte to be read from the input, pushes the byte to the top of the stack.

```
stack.push(read(stdin))
```

Write

- opcode is 0x09
- outputs the top of the stack as ASCII.

```
write(stdout, stack.pop())
```

LeT's rUN TogEaTHeR

Here you'll find an execution log of a simple program. Note that the ';' symbol starts a comment line lines of the form "; >| value1 value2 value3" show the stack state before the following instruction. The stack head is to the left (the first value after >| is SP[0]) The stack state inside the called function is a direct continuation of the caller execution
Note that "Word:" defines a label, which basically names a line of code.

```
;>|
   Push 2
;>| 02
   Push 7F
;>| 7F 02
   Read        ; assuming user inputs 0x3
;>| 03 7F 02
   Push 0A    ; OFFSET of Adder
;>| 0A 03 7F 02
   Call
;>| 82 02
   Divide
;>| 00 41
   Swap 1
;>| 41 00
   Write
;>| 00
   Pop
;>|
   Push 0C    ; OFFSET of More
;>| 0C
   JSE
;>|

NotReached:
   Push 4
   Push 0
   Sub         ; constructs offset of NotReached, which is -4 (0xFC)
   Call

Adder:
;>| 05 03 7F 02
   Load 2
;>| 7F 05 03 7F 02
   Load 2
;>| 03 7F 05 03 7F 02
   Add
```

```
;>| 82 05 03 7F 02
   Swap 3
;>| 7F 05 03 82 02
   Pop
;>| 05 03 82 02
   Swap 1
;>| 03 05 82 02
   Pop
;>| 05 82 02
   Ret
;>| 82 02

More:
; fill the rest on your own!
;>|
   Push 44
;>|
   Push 4E
;>|
   Push 45
;>|
   Push 20
;>|
   Write
;>|
   Write
;>|
   Write
;>|
   Write

; Program ends here
On the displayed run, The program printed “A END” Your job is to decipher the code and give us the flag.
Good Luck!
```

תוכן הקובץ שהורד:

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  BF C2 85 CB A2 CE 82 B2 E0 87 C9 A0 CC A0 CF 9D  ¿ ¯oo ‡È²,¯¢ … ¸.
00000010  FA 94 FD 91 FD 92 C0 87 E9 80 EC A0 CF 9D CF A0  ת” ‘ ’ À‡é€ì ÏÏ
00000020  F8 83 E4 85 E9 8F 8B 18 08 8C 11 41 8A 80 01 14  רƒä…é.‹....A.€..
00000030  B0 81 10 B1 09 AF 10 42 42 80 A5 14 42 21 80 A0  °..±.¯.BB€¥.B!€
00000040  14 80 21 44 9B 14 20 82 42 02 82 45 02 21 22 00  .€!D›. ‚B.‚E.!".
00000050  82 21 02 21 20 42 42 A4 80 01 11 82 03 00 22 20  ‚!.! BB¤€..‚.." 
00000060  20 23 20 21 20 12                                 # ! .
```

ההוראות למימוש המכונה הוירטואלית יחסית פשוטות, אפשר לממש בקלות עם סקריפט:

```python
import mmap, os, sys
import struct, re
import builtins
import ctypes

def memory_map(filename, access=mmap.ACCESS_WRITE):
    size = os.path.getsize(filename)
    fd = os.open(filename, os.O_RDWR)
    return mmap.mmap(fd, size, access=access)

OP_PUSH = 0x80
OP_LOAD = 0x40
OP_SWAP = 0x20
```

```
OP_ADD = 0x0
OP_SUB = 0x1
OP_MUL = 0x3
OP_DIV = 0x2
OP_JUMP = 0x10
OP_CALL = 0x11
OP_RET = 0x12
OP_CJE = 0x14
OP_JSE = 0x18
OP_READ = 0x08
OP_WRITE = 0x09

class Interpreter(object):
    def __init__(self):
        self.stack = []
        self.ip = 0
        self.debug = False
        self.num_reads = 0

    def log(self, s):
        if self.debug:
            print (s)

    def stack_index(self, index):
        return len(self.stack) - index - 1

    @staticmethod
    def signed_num(num):
        return ctypes.c_byte(num).value

    @staticmethod
    def unsigned_num(num):
        return ctypes.c_ubyte(num).value


    def execute_push(self, current_instruction):
        value = current_instruction - OP_PUSH
        self.log("Push {}".format(value))
        self.stack.append(value)

    def execute_load(self, current_instruction):
        value = current_instruction - OP_LOAD
        self.log("Load {}".format(value))
        self.stack.append(self.stack[self.stack_index(value)])

    def execute_swap(self, current_instruction):
        value = current_instruction - OP_SWAP
        if value == 0:
            self.log ("Pop")
            self.stack.pop()
        else:
            self.log("Swap {}".format(value))
            index = self.stack_index(value)
            temp = self.stack[index]
            self.stack[index] = self.stack.pop()
            self.stack.append(temp)

    def execute_add(self):
        self.log ("Add")
        self.stack.append(self.unsigned_num(self.signed_num(self.stack.pop()) +
self.signed_num(self.stack.pop())))

    def execute_sub(self):
        self.log ("Sub")
        self.stack.append(self.unsigned_num(self.signed_num(self.stack.pop()) -
self.signed_num(self.stack.pop())))
```

```
    def execute_mul(self):
        self.log ("Mul")
        self.stack.append( self.unsigned_num(
 (self.signed_num(self.stack.pop()) * self.signed_num(self.stack.pop())) % 256))

    def execute_div(self):
        self.log ("Div")
        a = self.stack.pop()
        b = self.stack.pop()
        self.stack.append(a // b)
        self.stack.append(a % b)

    def execute_jump(self):
        self.log ("Jump")
        self.ip = self.ip + self.signed_num(self.stack.pop())

    def execute_call(self):
        self.log ("Call")
        offset = self.stack.pop()
        self.stack.append(self.ip) # note that IP was already incremented here,
points to next instruction.
        self.ip = self.ip + self.signed_num(offset)
        #self.ip = self.ip + offset #Already signed?

    def execute_ret(self):
        self.log ("Ret")
        self.ip = self.stack.pop()

    def execute_cje(self):
        self.log ("CJE")
        offset = self.stack.pop()
        if self.stack.pop() == self.stack.pop():
            self.ip = self.ip + self.signed_num(offset)
            #self.ip = self.ip + offset #Already signed?

    def execute_jse(self):
        self.log ("JSE")
        offset = self.stack.pop()
        if len(self.stack) == 0:
            self.ip = self.ip + offset

    def execute_read(self):
        self.log ("Read")
        input_byte_str = input("Please input byte in format 0xab: ")
        input_byte = int(input_byte_str, 16)
        print (input_byte)
        self.stack.append(input_byte)
        self.num_reads += 1

    def execute_write(self):
        self.log ("Write")
        b = self.stack.pop()
        #sys.stdout.write(chr(b))
        print("\t --> \t'" + chr(b) + "'")

    def print_stack(self):
        if self.debug:
            sys.stdout.write(";>| ")
            for n in reversed(self.stack):
                sys.stdout.write("{:02X} ".format(n))
            sys.stdout.write("\n")

    def execute(self, path_to_code):
        self.code = memory_map(path_to_code, mmap.ACCESS_READ)
        while self.ip != len(self.code):
```

```
            #assert(self.num_reads <= 1)
            self.print_stack()
            current_instruction = self.code[self.ip]
            self.ip += 1
            if current_instruction & OP_PUSH:
                self.execute_push(current_instruction)
            elif current_instruction & OP_LOAD:
                self.execute_load(current_instruction)
            elif current_instruction & OP_SWAP:
                self.execute_swap(current_instruction)
            elif current_instruction == OP_ADD:
                self.execute_add()
            elif current_instruction == OP_SUB:
                self.execute_sub()
            elif current_instruction == OP_MUL:
                self.execute_mul()
            elif current_instruction == OP_DIV:
                self.execute_div()
            elif current_instruction == OP_JUMP:
                self.execute_jump()
            elif current_instruction == OP_CALL:
                self.execute_call()
            elif current_instruction == OP_RET:
                self.execute_ret()
            elif current_instruction == OP_CJE:
                self.execute_cje()
            elif current_instruction == OP_JSE:
                self.execute_jse()
            elif current_instruction == OP_READ:
                self.execute_read()
            elif current_instruction == OP_WRITE:
                self.execute_write()
```

נריץ את הקוד ונקבל:

```
Please input byte in format 0xab:
```

ננסה אקראית:

```
Please input byte in format 0xab: 0x00
0
     -->     '0'
```

נצטרך להבין טוב יותר מה בדיוק התוכנה רוצה. זמן לכתוב דיסאסמבלר בסיסי:

```
    def disassemble(self, path_to_code):
        self.code = memory_map(path_to_code, mmap.ACCESS_READ)
        for i in range(len(self.code)):
            current_instruction = self.code[i]
            sys.stdout.write("{:02X}\t{:02X}\t".format(i, current_instruction))
            if current_instruction & OP_PUSH:
                print("Push 0x{:02X}".format(current_instruction - OP_PUSH))
            elif current_instruction & OP_LOAD:
                print("Load 0x{:02X}".format(current_instruction - OP_LOAD))
            elif current_instruction & OP_SWAP:
                if current_instruction == OP_SWAP:
                    print("Pop")
                else:
                    print("Swap 0x{:02X}".format(current_instruction - OP_SWAP))
            elif current_instruction == OP_ADD:
                print("Add")
            elif current_instruction == OP_SUB:
                print("Sub")
            elif current_instruction == OP_MUL:
                print("Mul")
            elif current_instruction == OP_DIV:
```

```
            print("Divide")
        elif current_instruction == OP_JUMP:
            print("Jump")
        elif current_instruction == OP_CALL:
            print("Call")
        elif current_instruction == OP_RET:
            print("Ret")
        elif current_instruction == OP_CJE:
            print("CJE")
        elif current_instruction == OP_JSE:
            print("JSE")
        elif current_instruction == OP_READ:
            print("Read")
        elif current_instruction == OP_WRITE:
            print("Write")
```

נריץ ונקבל:

```
00  BF  Push 0x3F
01  C2  Push 0x42
02  85  Push 0x05
03  CB  Push 0x4B
04  A2  Push 0x22
05  CE  Push 0x4E
06  82  Push 0x02
07  B2  Push 0x32
08  E0  Push 0x60
09  87  Push 0x07
0A  C9  Push 0x49
0B  A0  Push 0x20
0C  CC  Push 0x4C
0D  A0  Push 0x20
0E  CF  Push 0x4F
0F  9D  Push 0x1D
10  FA  Push 0x7A
11  94  Push 0x14
12  FD  Push 0x7D
13  91  Push 0x11
14  FD  Push 0x7D
15  92  Push 0x12
16  C0  Push 0x40
17  87  Push 0x07
18  E9  Push 0x69
19  80  Push 0x00
1A  EC  Push 0x6C
1B  A0  Push 0x20
1C  CF  Push 0x4F
1D  9D  Push 0x1D
1E  CF  Push 0x4F
1F  A0  Push 0x20
20  F8  Push 0x78
21  83  Push 0x03
22  E4  Push 0x64
23  85  Push 0x05
24  E9  Push 0x69
25  8F  Push 0x0F
26  8B  Push 0x0B
27  18  JSE
28  08  Read
29  8C  Push 0x0C
2A  11  Call
2B  41  Load 0x01
2C  8A  Push 0x0A
2D  80  Push 0x00
2E  01  Sub
2F  14  CJE
```

```
30  B0  Push 0x30
31  81  Push 0x01
32  10  Jump
33  B1  Push 0x31
34  09  Write
35  AF  Push 0x2F
36  10  Jump
37  42  Load 0x02
38  42  Load 0x02
39  80  Push 0x00
3A  A5  Push 0x25
3B  14  CJE
3C  42  Load 0x02
3D  21  Swap 0x01
3E  80  Push 0x00
3F  A0  Push 0x20
40  14  CJE
41  80  Push 0x00
42  21  Swap 0x01
43  44  Load 0x04
44  9B  Push 0x1B
45  14  CJE
46  20  Pop
47  82  Push 0x02
48  42  Load 0x02
49  02  Divide
4A  82  Push 0x02
4B  45  Load 0x05
4C  02  Divide
4D  21  Swap 0x01
4E  22  Swap 0x02
4F  00  Add
50  82  Push 0x02
51  21  Swap 0x01
52  02  Divide
53  21  Swap 0x01
54  20  Pop
55  42  Load 0x02
56  42  Load 0x02
57  A4  Push 0x24
58  80  Push 0x00
59  01  Sub
5A  11  Call
5B  82  Push 0x02
5C  03  Mul
5D  00  Add
5E  22  Swap 0x02
5F  20  Pop
60  20  Pop
61  23  Swap 0x03
62  20  Pop
63  21  Swap 0x01
64  20  Pop
65  12  Ret
```

החלק הראשון בסך הכל מכין את המחסנית להרצה, ואפשר לדלג עליו לעת עתה.

כך נראה החלק השני, אחרי הוספת הערות:

```
label4:
26      8B      Push 0x0B
27      18      JSE        ; to label1 - jumps only when one value is left on the stack
28      08      Read
29      8C      Push 0x0C
2A      11      Call       ; to label2
2B      41      Load 0x01
2C      8A      Push 0x0A
2D      80      Push 0x00
2E      01      Sub
2F      14      CJE        ; to label4
30      B0      Push 0x30
31      81      Push 0x01
32      10      Jump       ; to label5
label1:
33      B1      Push 0x31
label5:
34      09      Write
35      AF      Push 0x2F
36      10      Jump       ; to label6
label2:
37      42      Load 0x02  ; Take top of payload
38      42      Load 0x02  ; Take input
39      80      Push 0x00
3A      A5      Push 0x25  ; Address of label3
3B      14      CJE        ; to label3 ; Jump to label3 if input == 0
3C      42      Load 0x02  ; Take input
3D      21      Swap 0x01  ; Take top of payload
3E      80      Push 0x00  ;
3F      A0      Push 0x20  ; Address of label3
40      14      CJE        ; to label3 ; Jump to label3 if top of payload == 0
41      80      Push 0x00
42      21      Swap 0x01  ; Take input
43      44      Load 0x04  ; Take top of payload
44      9B      Push 0x1B  ; Address of label3
45      14      CJE        ; to label3 ; Jump to label3 if input == top of payload
46      20      Pop        ; Pop 0
47      82      Push 0x02
48      42      Load 0x02  ; Take input
49      02      Div        ; input / 2
4A      82      Push 0x02
4B      45      Load 0x05  ; Take top of payload
4C      02      Div        ; Top of payload  / 2
4D      21      Swap 0x01  ; Take div(top of payload / 2)
4E      22      Swap 0x02  ; Take mod(input / 2)
4F      00      Add        ; div(top of payload / 2) + mod(input / 2)
50      82      Push 0x02
51      21      Swap 0x01  ; Take div(top of payload / 2) + mod(input / 2)
52      02      Div        ; ( div(top of payload / 2) + mod(input / 2) ) / 2
53      21      Swap 0x01  ; Take div( div(top of payload / 2) + mod(input / 2) ) / 2
54      20      Pop        ; Ignore div( div(top of payload / 2) + mod(input / 2) ) / 2, take
                           ; mod( div(top of payload / 2) + mod(input / 2) ) / 2
55      42      Load 0x02  ; Take div (input / 2)
56      42      Load 0x02  ; Take div (top of payload / 2)
57      A4      Push 0x24
58      80      Push 0x00
59      01      Sub        ; Offset of label2
5A      11      Call       ; to label2
5B      82      Push 0x02
5C      03      Mul        ; ret * 2
5D      00      Add        ; (ret * 2) + (mod( mod(top of payload / 2) + mod(input / 2) ) / 2)
5E      22      Swap 0x02  ; Take div(input / 2)
5F      20      Pop        ;
60      20      Pop        ; Stack: div (top of payload / 2) * 2
```

```
label3:                     ;
61      23      Swap 0x03   ;
62      20      Pop         ;
63      21      Swap 0x01   ;
64      20      Pop         ;
label6:                     ;
65      12      Ret         ; Return param[0]?
```

כעת אפשר לנסות לשחזר את הקוד בשפה עילית:

```
def unknown_function(input, top_of_payload):
    if input == 0:
        return top_of_payload
    elif top_of_payload == 0:
        return input
    elif top_of_payload == input:
        return 0

    temp = unknown_function(top_of_payload // 2, input // 2)
    temp *= 2
    temp += (( (top_of_payload % 2) + (input % 2) ) % 2)
    return temp
```

הקוד הזה פועל על ראש המחסנית, ומשווה את הקלט מהמשתמש אל הערך ששמור על המחסנית. אם הערך נכון, ממשיכים לאיטרציה הבאה, ואם לא - יוצאים.

לכן, כדי לדעת מה הקלט שהתוכנה מצפה לו, נעתיק את תוכן המחסנית ונריץ Brute Force על כל האפשרויות:

```
stack = "0F 69 05 64 03 78 20 4F 1D 4F 20 6C 00 69 07 40 12 7D 11 7D 14 7A 1D 4F
20 4C 20 49 07 60 32 02 4E 22 4B 05 42 3F".split(" ")
stack = [int(x, 16) for x in stack]
stack.reverse()

while len(stack) > 1:
    top_of_stack = stack.pop()
    for i in range(256):
        if unknown_function(i, top_of_stack) == stack[-1]:
            sys.stdout.write(chr(i))
            break
```

התוצאה היא:

```
flag{XoRRoLlinGRollingRolliNgR0LliNG}
```

אתגר 11: Bowsers Secret Message (קטגוריית Reversing, 85 נקודות)

תיאור האתגר:

We uncovered Bowser's old laptop!

Everything was wiped except for 3 files, he must have used them to send his evil henchmen --**Steganos & Graphein** -- a secret message.

Help us!

הקבצים המצורפים הם:

1. Secret.gif - קובץ GIF מונפש שנפתח ללא בעיה
2. Enc.py - סקריפט להצפנת מסר סודי בתוך קובץ GIF
3. Lzwlib.py - סקריפט עזר לביצוע דחיסה

תוכן הקובץ enc.py:

```
from __future__ import print_function
from random import randint, shuffle
import sys
from struct import unpack, pack as pk
from io import BytesIO as BIO
import lzwlib

up = lambda *args: unpack(*args)[0]


def F(f):
    assert f.read(3) == 'GIF', ''
    assert f.read(3) == '89a', ''
    w, h = unpack('HH', f.read(4))

    assert 32 <= w <= 500, ''
    assert 32 <= h <= 500, ''
    logflags = up('B', f.read(1))

    assert logflags & 0x80, ''
    size_count = logflags & 0x07

    gct_count = 2**(size_count+1)
    assert 4 <= gct_count <= 256, ''

    bgcoloridx = up('B', f.read(1))
    f.seek(1, 1)
    clrs = []
    for i in xrange(gct_count):
        clr = (up('B', f.read(1)), up('B', f.read(1)), up('B', f.read(1)))
        clrs.append(clr)

    assert len(clrs) > bgcoloridx, ''
    return clrs, bgcoloridx, size_count, h, w


class T(object):
    I = 0
    EG = 1
    EA = 2
    EC = 3
    ET = 4


def C(f):
    rb = f.read(1)
    b = up('B', rb)
```

```
    while b != 0x3B:
        buf = ''
        buf += rb
        if b == 0x2c:
            nbuf = f.read(2*4)
            eb = f.read(1)
            assert (up('B', eb) & 0x03) == 0, ''
            nbuf += eb

            nbuf += f.read(1)
            nbuf += V(f)
            t = T.I
        elif b == 0x21:
            rb = f.read(1)
            buf += rb
            b = up('B', rb)

            if b == 0xF9:
                nbuf = f.read(1)
                blksize = up('B', nbuf)
                nbuf += f.read(blksize)
                nbuf += f.read(1)
                assert nbuf[-1] == '\x00', ''
                t = T.EG
            elif b in [0xFF, 0x01]:
                nbuf = f.read(1)
                blksize = up('B', nbuf)
                nbuf += f.read(blksize)
                nbuf += V(f)

                t = (b+3) & 0x0F
            elif b == 0xFE:
                nbuf = V(f)

                t = T.EC
            else:
                raise Exception("unsupprted thing @{}".format(f.tell()))

        buf += nbuf

        yield t, buf
        rb = f.read(1)
        b = up('B', rb)

    yield None, '\x3B'

    raise StopIteration


def WB(buf):
    blockcount = len(buf)/0xFF
    blockcount += 1 if len(buf) % 0xFF else 0

    blocks = [
        pk('B', len(subblock))+subblock for subblock in [
            buf[i:0xFF+i] for i in xrange(0, blockcount*0xFF, 0xFF)
        ]
    ]

    return ''.join(blocks) + '\x00'


def k(bf):
    combined_buf = ''
    while True:
        cb = ord(bf.read(1))
        if not cb:
            break

        combined_buf += bf.read(cb)
    return combined_buf
```

```
def V(f):
    sbx = ''
    while True:
        rcb = f.read(1)
        sbx += rcb
        if rcb == '\x00':
            break

        cb = up('B', rcb)
        blk = f.read(cb)
        sbx += blk

    return sbx


def Q(delay, w, h, x, y, tidx):

    assert 0 <= tidx <= 255
    assert 0 <= delay < 2**16

    indices = [tidx]*(w*h)
    buf = BIO('')

    buf.write('\x21\xF9\x04\x05')
    buf.write(pk('H', delay))
    buf.write(pk('B', tidx))
    buf.write('\x00')

    buf.write('\x2c')
    buf.write(pk('H', x))
    buf.write(pk('H', y))
    buf.write(pk('H', w))
    buf.write(pk('H', h))
    buf.write('\x00')

    LZWMinimumCodeSize = 8

    cmprs,    = lzwlib.Lzwg.compress(
        indices, LZWMinimumCodeSize)

    obuf = pk('B', LZWMinimumCodeSize) + WB(cmprs)

    buf.write(obuf)
    buf.seek(0)
    return buf.read()


def z(n):
    import math
    for i in xrange(1, int(math.sqrt(n) + 1)):
        if n % i == 0:
            yield i


def m(a, mm, hh):
    if a < 0x08:
        if a % 2:
            _0 = 0
            _1 = 1
             4 = a << 2
             3 = randint(4, mm-1)
        else:
            _0 = 1
            _1 = a << 1
            _3 = randint(4, mm/2)
             4 = randint(4, hh/3)
    else:
        ds = list(z(a))
        _0 = 0
        _1 = 0
        shuffle(ds)
         4 = ds[0]
        assert a % _4 == 0
        _3 = a/_4
```

```
    return _0, _1, _3, _4


def h(b6, b1, mw, mh, mci, d=3):
    idx = randint(0, (mci-1)/2)*2 + b1
    x, xx, xxx, xxxx = m(b6, mw, mh)
    f = Q(d, xxx, xxxx, x, xx, idx)
    return f


def M(s):
    l = list(set(s.upper()))
    shuffle(l)
    d = ''.join(l)
    assert len(d) <= 2**6, ''
    return d, [(d.index(c.upper()), int(c.isupper())) for c in s]


def E(f, s, o):
    global_colors, bgcoloridx, size_count, hh, ww = F(f)
    mp, ks = M(s)
    hdr_end = f.tell()
    f.seek(0)
    o.write(f.read(hdr_end))
    fc = 0

    o.write('\x21\xFE')
    o.write(WB('RDBNB'+mp))
    o.flush()

    for t, buf in C(f):
        print('.', end='')
        if t == T.EC:
            continue
        if t == T.EG:
            if ks:
                delay = up('<H', buf[4:6])
                assert delay >= 6
                buf = buf[:4] + pk('<H', delay - 3) + buf[6:]
            obuf = buf

        elif t == T.I:
            fc += 1
            total_raw_blocks_data = ''
            bf = BIO(buf)
            pref = bf.read(10)

            LZWMinimumCodeSize = ord(bf.read(1))
            total_raw_blocks_data = k(bf)

            indices, dcmprsdcodes = lzwlib.Lzwg.decompress(
                total_raw_blocks_data, LZWMinimumCodeSize)
            xxx = unpack('<B H H H H B', pref)

            cmprs, codes = lzwlib.Lzwg.compress(
                indices, LZWMinimumCodeSize)

            obuf = pref + pk('B', LZWMinimumCodeSize) + WB(cmprs)

            if ks:
                mpindx, isup = ks.pop(0)
                obuf += h(mpindx, isup,
                          ww, hh, len(global_colors)-1)
        else:
            obuf = buf

        o.write(obuf)
    o.flush()
    assert not ks, ''

    return 0
```

```
if __name__ == '__main__':
    assert len(sys.argv) > 2, 'bad input'
    fpath = sys.argv[1]
    flag = sys.argv[2]
    if len(sys.argv) > 3:
        outpath = sys.argv[3]
    else:
        outpath = fpath + '.out.gif'

    f = open(fpath, 'rb')
    o = open(outpath, 'wb')
    rv = E(f, flag, o)
    sys.exit(rv)
```

ניתן לראות שהדגל מתקבל כפרמטר לפונקציית E, שבתורה שולחת אותו ל-M:

```
mp, ks = M(s)
```

פונקציית M מייצרת ייצוג אחר של הדגל:

```
def M(s):
    l = list(set(s.upper()))
    shuffle(l)
    d = ''.join(l)
    assert len(d) <= 2**6, ''
    return d, [(d.index(c.upper()), int(c.isupper())) for c in s]
```

הדגל מיוצג בתור set של האותיות בדגל, יחד עם מערך של זוגות שמכילים שני נתונים אודות כל אות בדגל: מהו מיקומה של האות ב-set, והאם זו אות גדולה או קטנה.

את ה-set והמערך יחביאו במקומות שונים בתוך התמונה. את ה-set (שנקרא mp) קל למצוא:

```
    o.write('\x21\xFE')
    o.write(WB('RDBNB'+mp))
    o.flush()
```

ובתוכן התמונה:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 21 FE 19  .............! .
52 44 42 4E 42 7B 55 46 4B 52 41 59 57 53 7D 54  RDBNB{UFKRAYWS}T
4C 4D 47 49 21 45 4F 5F 48 00 21 FF 0B 4E 45 54  LMGI!EO_H.!□.NET
53 43 41 50 45 32 2E 30 03 01 00 00 00 21 F9 04  SCAPE2.0.....!ω.
```

כלומר, ה-set שלנו הוא:

```
{UFKRAYWS}TLMGI!EO_H
```

כעת נשאר לפצח את מיקום המערך. המערך (ks) מתחיל את תהליך ההצפנה בקוד הבא:

```
if ks:
   mpindx, isup = ks.pop(0)
   obuf += h(mpindx, isup, ww, hh, len(global_colors)-1)
```

הוא מפורק לשני הגורמים שלו (mpindex, isup) ונשלח לפונקציית h.

```
def h(b6, b1, mw, mh, mci, d=3):
    idx = randint(0, (mci-1)/2)*2 + b1
    x, xx, xxx, xxxx = m(b6, mw, mh)
    f = Q(d, xxx, xxxx, x, xx, idx)
    return f
```

- isup, שהוא בוליאני, מוסתר בתוך הזוגיות של idx (זוגי -> אות קטנה, אי-זוגי -> אות גדולה).
- mpindex נשלח ל-m והתוצאה נשלחת ל-Q.

פונקציית m מסתירה את הערך באמצעות מספר מניפולציות, ואז פונקציית Q כותבת את התוצאה אל תוך התמונה מיד אחרי magic number:

```
buf.write('\x21\xF9\x04\x05')
```

לכן, כדי למצוא את הערכים המקוריים, פשוט נפעיל לוגיקה הפוכה לזאת שהפעילה m:

```
import struct, sys

s = "{UFKRAYWS}TLMGI!EO_H"

with open(r"secret.gif", "rb") as f:
    content = f.read()
    location = 0
    indices = []
    while True:
        location = content.find(b'\x21\xF9\x04\x05', location+1)
        if location == -1:
            break

        (header, delay, tidx, zero, h2c, x, y, w, h, zero2) =
struct.unpack_from(b'<IHBBBHHHHB', content, location)
        print ("{:02X}, {:02X}, {:02X}, {:02X}".format(x, y, w, h))
        #print ("{:02X}, {:02X}, {:02X}, {:02X}, {:02X}, {:02X}, {:02X}, {:02X},
{:02X}, {:02X}".format(header, delay, tidx, zero, h2c, x, y, w, h, zero2))
        if x == 0 and y == 0:
            mpindx = w * h
        elif x == 0 and y == 1:
            mpindx = h >> 2
        elif x == 1:
            mpindx = y >> 1
        else:
            print ("!!!!")
        print (mpindx)
        indices.append((mpindx, tidx % 2 == 0))

        print ("\n")

    for mpindx, isup in indices:
        c = s[mpindx] if not isup else s[mpindx].lower()
        sys.stdout.write(c)
```

נריץ ונקבל:

```
flag{thIs_mushRooM_w!lL_maKe_you_tAlLer}
```

אתגר 12: Trace me if you Can (קטגוריית Surprise, 150 נקודות)

תיאור האתגר:

Hi There,
We've been working on this one for a while now. This machine spits out the flag when given the right input. We're not sure what the input should be but we managed to get this weird looking trace. Our best minds spent days, but we still can't figure it out!
35.194.63.219:2005
Please help us

הקישור לקובץ הכיל Trace בייצוג ביניים מסוג SSA (Static single assignment form). מיד נסביר מה זה אומר, אבל לפני הכל - באתגר הזה התשובה שלי לא התקבלה. לכן, אני אסביר את השלבים שעברתי, אבל עדיין יהיה חסר פה גרוש לשקל. עוד אודה שבמהלך האתגר התייעצתי עם YaakovCohen88 מ-JCTF.

כאשר מקמפלים תוכנה, הקומפיילר יכול לעבור דרך מספר שלבי ביניים עד שהוא מגיע אל התוצאה הסופית שלו (למשל: קוד מכונה). SSA הוא שלב-ביניים כזה, שמתאפיין בכך שכל משתנה מקבל השמה פעם אחת בלבד, וכל משתנה מוגדר לפני שמשתמשים בו.

חשוב לשים לב שהקובץ הכיל Trace של ריצה בפורמט הזה, ולא ייצוג של קומפילציה של תוכנה ב-SSA. כלומר, לא קיבלנו תוכנה מלאה, אלא ריצה מסויימת של תוכנה על קלט מסויים, כשבפועל אנחנו נחשפים אך ורק לפקודות שאכן רצו בריצה המסויימת הזו. איננו נחשפים לפונקציות שלא נקראו, תנאים שלא התקיימו וכו'.

הקובץ הכיל כ-190,000 שורות, אך לשם הדוגמא נצרף קטע קצר (ההערות במקור):

```
Starting main.kendrick at
/home/gull.omer/go/src/omer/ssa/omerr/version1/ssa.go:238:6.
.0:
     t0 = len(a)
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.2:
     t5 = &a[t3]
     t6 = *t5
     t7 = t1 + t6
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.3:
     return t1
Returning from main.kendrick, proceeding main.main at
/home/gull.omer/go/src/omer/ssa/omerr/version1/ssa.go:306:21.
```

הקטע הזה מייצג את הריצה של פונקציה בשם Kendrick. היא מקבלת משתנה בשם a, ומתייחסת אליו כמערך.

לאורך הקוד, ניתן לראות הגדרות של "תוויות" כגון ".0:", ".1:" וכו'. תוויות אלו משמשות לניהול הריצה של הפונקציה. למשל, ניהול של לולאה יכלול בדיקת תנאי ואז קפיצה אל תווית של תוכן הלולאה במקרה אחד, או קפיצה אל התווית של הלוגיקה שאחרי הלולאה במקרה אחר. למעשה, זה בדיוק מה שאנחנו רואים ב-Kendrick - התווית "1" היא בדיקת תנאי הלולאה, התווית "2" היא תוכן הלולאה והתווית "3" היא הלוגיקה שאחרי הלולאה. בריצה הזו, הלולאה רצה פעם אחת באופן מלא ואז סיימה את פעולתה.

בתרגום ל-Python, הפונקציה שקולה (כנראה) ל:

```
def kendrick(a): #sum
    damn = 0
    t2 = -1
    while (t2 + 1 < len(a)):
        damn += a[t2 + 1]
        t2 += 1
    return damn
```

על מנת לתרגם את הפונקציה, עבדתי בשיטה איטרטיבית של צמצום.

השלב הראשון:

```
.0:
     t0 = len(a)
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.2:
     t5 = &a[t3]
     t6 = *t5
     t7 = t1 + t6
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < t0
     if t4 goto 2 else 3
.3:
     return t1
```

בשלב הבא:

```
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < len(a)
     if t4 goto 2 else 3
.2:
     t6 = *&a[t3]
     t7 = t1 + t6
     jump 1
.1:
     t1 = phi [0: 0:int, 2: t7] #damn
     t2 = phi [0: -1:int, 2: t3]
     t3 = t2 + 1:int
     t4 = t3 < len(a)
     if t4 goto 2 else 3
.3:
     return t1
```

משתנים מסוג phi הם כאלה שמקבלים לפעמים ערך אחד ולפעמים ערך אחר. במקרים רבים הערך הראשון הוא ערך התחלתי והערך השני הוא ערך הריצה.

לכן:

```
damn = 0 # a.k.a. t1, a.k.a. t7
i = -1 # a.k.a. t2, a.k.a. t3
.1:
        i = i+ 1
     if i < len(a) goto 2 else 3
.2:
     damn = damn + a[i]
     jump 1
.1:
        i = i+ 1
     if i < len(a) goto 2 else 3
.3:
     return t1
```

ומשם לא קשה להגיע ללולאה שראינו ב-Python. באמצעות הלוגיקה הזו, תרגמתי את ה-Trace כולו:

```
class TraceException(Exception):
    pass

def guru(a, b): # Max
    if a > b:
        return a
    else:
        return b

def andre(a, b): # Multiply
    t10 = [0] * (len(a) + len(b))

    for i in range(len(b)):
        for j in range(len(a)):
```

```
            t10[i + j] += (a[j] * b[i])

    t21 = len(t10[:len(t10) - 1])
    for i in range(t21):
        if (t10[i] >= 10):
            t10[i + 1] += (t10[i] // 10)
            t10[i] = t10[i] % 10

    return t10

def rakim(a, b):
    if (doom(a, b) == -1):
        raise TraceException("1")
    else:
        t5 = [None] * len(a)
        for i in range(len(a)):
            a_i = 0
            b_i = 0
            #3
            if i < len(a):
                #6
                a_i = a[i]
            #7
            if i < len(b):
                #8
                b_i = b[i]
            #9
            if a_i < b_i:
                #10
                raise TraceException("2")
            else:
                #11
                t5[i] = a_i - b_i


        #4
        return t5

def doom(aa, bb): # Compare
    """
    ii = len(aa) // 2
    while(ii >= 1):
        if aa[ii] == 0:
            aa[ii] = 0
        else:
            ii -= 1
    """

    i = len(aa) - 1
    m = []
    while (i >= 0):
        if aa[i] > 0:
            m = aa[:i + 1]
            break
        else:
            i -= 1

    i = len(bb) - 1
    f = []
```

```
    while (i >= 0):
        if (bb[i] > 0):
            f = bb[:i + 1]
            break
        else:
            i -= 1

    """
    i = len(aa) - 1
    while(i >= 0):
        if aa[i] > 0:
            t43 = 0 + aa[i]
    """

    if (len(m) > len(f)):
        return 1
    if (len(m) < len(f)):
        return -1


    i = len(m) - 1
    #27
    while(i >= 0):
        #25
        if m[i] > f[i]:
            #28
            raise TraceException("3")
        else:
            #29
            if m[i] < m[i]:
                #30
                raise TraceException("4")
            else:
                #31
                i -= 1
    #26
    return -2

def gza(a, b): # Add
    t2 = guru(len(a), len(b))
    t4 = [None] * (t2 + 1)
    r = rakim(a, [0])
    d = doom(a, r)
    if d != -2:
        raise TraceException("5")

    wu = 0
    for i in range(len(t4)):
        #4
        if i < len(a):
            #6
            t19 = a[i]
        else:
            t19 = 0 # ?
        #7
        a_i = t19
        if (i < len(b)):
            #8
            t24 = b[i]
```

```
        else:
            t24 = 0 # ?
        #9
        b_i = t24
        t27 = a_i + b_i + wu
        wu = t27 // 10
        if t27 >= 10:
            #10
            raise TraceException("6")
        else:
            #11
            tmp = t27
            t4[i] = tmp
    #5
    return t4



def nas(a): # ATOI

    #t5 = [0] * ( (len(a) // 2) + (len(a) // 2) )
    t5 = []

    z = 0
    msg = t5
    i = len(a) - 1
    while(i >= 0):
        t10 = ord(a[i]) - 48
        if (t10 >= 0) and (t10 < 10):
            z += t10
            msg.append(guru(t10, 0))

        if (z == 1024):
            raise TraceException ("7")
        else:
            i -= 1

    return msg


def kendrick(a): # Sum
    damn = 0
    t2 = -1
    while (t2 + 1 < len(a)):
        damn += a[t2 + 1]
        t2 += 1
    return damn

##################

def main(my_input):

    t8 = my_input
    t11 = t8.split(" ")
    print ("Input length: {}".format(len(t11)))
    t13 = [None] * len(t11)

    for i in range(len(t11)):
        t13[i] = int(t11[i])
```

```
    t27 = andre([1], [0, 1])          # 10 * 1
    t32 = gza([0, 2], t27)            # + 20
    t37 = gza([0, 2], t32)            # + 20
    t42 = andre([guru(2, 1)], t37)    # * 2
    t45 = nas(str(len(t13)))          #
    t46 = doom(t45, t42)              # == 100?
    if t46 != -2:
        raise TraceException ("8 - Input Length != 100")

    for i in range(len(t13)):
        if t13[i] <= 0:
            raise TraceException("9 - Non-Positive Member!")

    t58 = [None] * (len(t13) // 2)

    for i in range(len(t58)):
        t58[i] = t13[i * 2] - t13[(i * 2) + 1]

    for i in range(len(t58)):
        t80 = nas(str(kendrick(t58[:i + 1])))
        t84 = doom(t80, [0])
        if t84 == -1:
            raise TraceException("10")

    o_su = 0
    e_su = 0
    for i in range(0, len(t13) - 1, 2):
        e_su +=  t13[i]
        o_su +=  t13[i + 1]

    if o_su != e_su:
        raise TraceException("11 - o_su != e_su ({} != {})".format(o_su,
e_su))

    u_arr = []
    x = 0
    for i in range(len(t13)):
        if x < len(u_arr):
            raise TraceException("12")


    if t13[49] != 102:
        raise TraceException("13 - t13[49] != 102")

    if t13[4] - t13[5] != t13[8]:
        raise TraceException("14 - t13[4] - t13[5] != t13[8]")


if __name__ == "__main__":
    input_arr = ( [1, 1, 1, 1, 5, 2, 1, 1, 3] + ([1] * 40) + [102, 102]
+ ([1] * 47) + [1, 6] )
    input_str = ' '.join([str(num) for num in input_arr])
    print (input_str)

    main(input_str)
```

נקודות ראויות לציון:

- התוכנה מקבלת קלט של מחרוזת ארוכה, מפצלת אותו לפי רווחים, מתייחסת לכל איבר כמספר ובודקת שמערך המספרים מתאים לכללים שהוגדרו מראש.
- כאשר אחד הכללים מופר, התוכנה המקורית הרימה דגל. במימוש שלי זרקתי Exception. כמו כן, בחרתי לזרוק Exception כאשר הלוגיקה הייתה חסרה מה-Trace (למשל: תנאי שמעולם לא התממש בריצה).
- התוכנה מייצגת מספרים בתור מערך הפוך. למשל, המספר 10 מיוצג בתור [0, 1]. פונקציות העזר של התוכנה משמשות לביצוע פעולות מתמטיות בסיסיות על המספרים בייצוג זה, כדוגמת חיבור, כפל והשוואה.
- לעיתים התוכנה ביצעה פעולות שאין להן משמעות, כדוגמת קטע הקוד בתחילת פונקציית doom (במקרה זה למשל השארתי את הקוד כהערה).
- הכללים שהתוכנה אוכפת:
  - מספר האיברים בקלט צריך להיות 100
  - כל האיברים צריכים להיות חיוביים
  - סכום האיברים במקומות הזוגיים והאי-זוגיים צריך להיות שווה
  - האיבר ה-49 צריך להיות 102
  - האיבר הרביעי פחות האיבר החמישי צריך להיות שווה לאיבר ה-13
  - שאר התנאים שקיימים ב-main מתממשים תמיד, בין השאר עקב מגבלות המימוש של פונקציות העזר (לפי מיטב הבנתי)

בפועל, למרבה הצער, השרת של האתגר לא הסכים לקבל קלטים שצייתו לכללים הללו. בשלב מסוים ייצרתי KeyGen בתקווה לייצר שונות כלשהי בקלטים שאני מנסה, למקרה שמשהו לא תקין בקלט הספציפי שייצרתי ידנית, אולם השרת המשיך לדחות את התשובה. לכן כנראה שחסר תנאי כלשהו, או קיימת בעיה כלשהי במימוש שלי. ואף על פי כן, הלוגיקה שמשתקפת ב-Reversing של ה-Trace מסתדרת עם סוג האתגר ומתאימה יחדיו בצורה יחסית טובה, ולכן אני מאמין שלא הייתי רחוק מדי מהתשובה הנכונה.

מילות סיכום

בסך הכל, אני חייב להודות שנהניתי מאוד מהאתגר, ואני שמח לראות שהטרנד של אתגרים כחול-לבן תופס תאוצה בשנים האחרונות. האתגר הזה כלל מספר רב יחסית של שלבים במגוון נושאים שונים וברמות קושי שונות, כך שהוא התאים הן למתחילים והן למשתתפים מנוסים יותר.

השלב המהנה ביותר בעיני היה שלב הפאזל, אולי כי במהלך ה-Day Job שלי (Embedded C) נוהגים שלא להשתמש ברקורסיה (מחשש שהמחסנית תגמר) ואולי כי יש משהו קסום בפתרון של צעד בודד שמצליח להתמודד עם בעיה שלמה. אני חושב שאני עדיין קצת מופתע כשזה פשוט עובד.

גם שאר השלבים המתקדמים יותר (המילון, המכונה הווירטואלית והסטגנוגרפיה) הצריכו השקעה יפה.

מנגד, האכזבה הגדולה ביותר הייתה שלב הפינג-פונג, קיוויתי שהוא יכלול יותר מאשר לשלוח חזרה מספר שהתקבל מהשרת.

בשלב האחרון באתגר ("Trace me if you can") השקעתי שעות על גבי שעות, אך לצערי לא הצלחתי לפתור אותו, וכך גם שמעתי מאחרים בתחום. יהיה מעניין לראות אם מישהו הצליח לפתור אותו (אני מניח שכן) וכמה רחוק הפתרון היה מהלוגיקה שהגעתי אליה.

אני מאוד מקווה להמשיך לראות אתגרים דומים בשנים הקרובות, כל הכבוד ליוצרים על אתגר מהנה ומושקע.