152 – Find video and time info.

Team Information

Team Name: kimbabasaksaksak

Team Member: Jaeheon Kim, Donghyun Kim, Soyoung Yoo, Minhee Lee

Email Address: uaaoong@gmail.com

Instructions

Description A movie file, but there is an unknown file that does not play normally in the playback viewer. Video, voice, and time information are recorded in this file, and many books are recorded in the video file. We need to restore the video and find the time information.

Target	Hash (MD5)
problem-final.img	CB91D34E37E31561079909945B4E287E

Questions

1. Submit a book title for a video file with different time information recorded between the video and the video frame data. (50 points)
2. Submit the title of the book containing recorded video time with time information from the year 2024 (The time information of the video is located at the beginning of the video data). (100 points)

Teams must:

- Describe step-by-step processes for generating your solution.
- Specify any tools used for this problem.

Tools used:

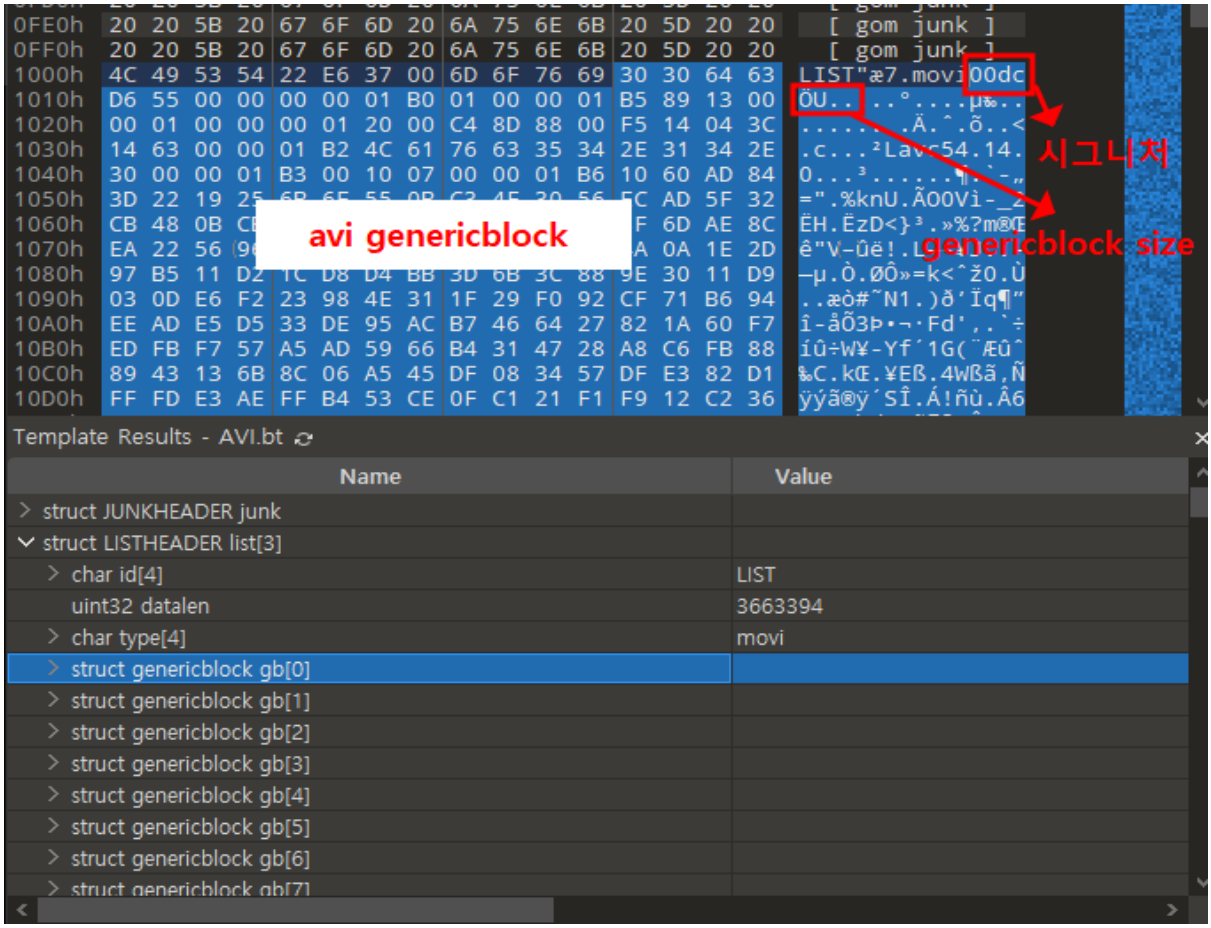
Name:	010 Editor	Publisher:	SweetScape
Version:	13.0.2		
URL:	https://www.sweetscape.com		

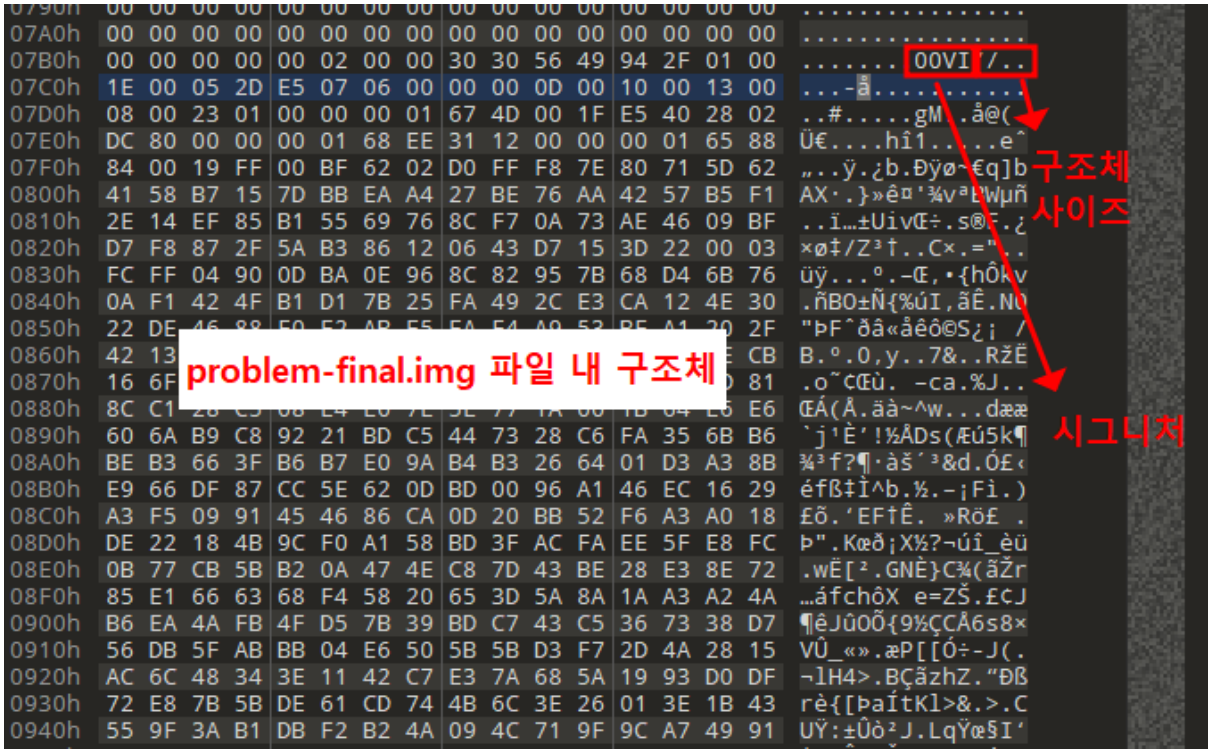
Name:	ffmpeg	Publisher:	
Version:	4.2.7-0ubuntu0.1		
URL:	https://ffmpeg.org		

Step-by-step methodology:

Q1. Submit a book title for a video file with different time information recorded between the video and the video frame data. (50 points)

문제에서 주어진 problem-final.img 파일을 확인하면, avi 컨테이너의 genericblock과 유사한 구조체가 존재한다. avi 컨테이너에서 genericblock은 시그니처마다 의미하는 데이터가 지정되어 있다. 00dc 시그니처는 비디오 프레임 데이터, 01wb는 오디오 데이터를 의미한다.

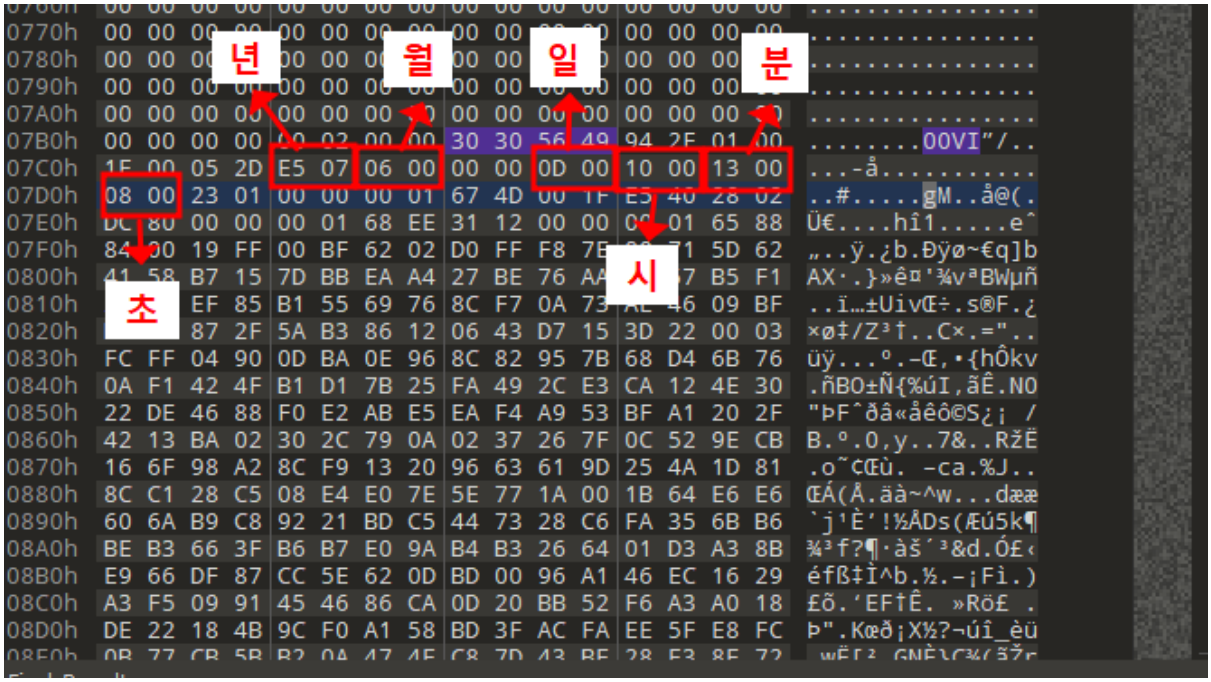




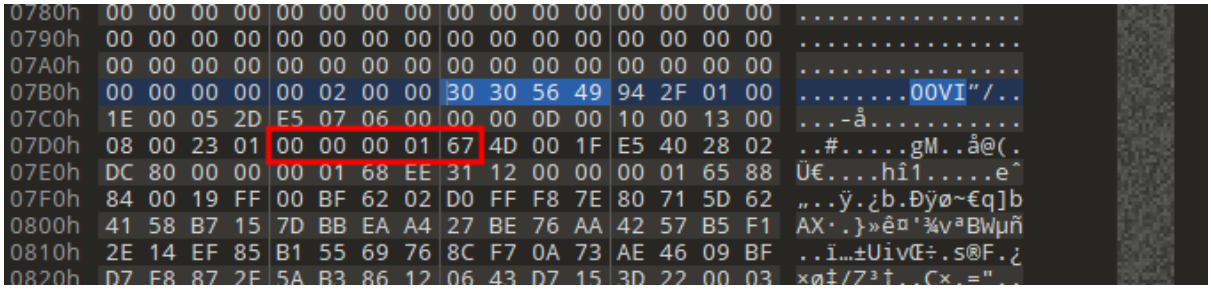
problem-final.img 파일 내 존재하는 구조체들의 모든 시그니처를 확인한 결과, 00VI, 01VI, 00VP, 01VP, 00AD, 01AD, 00SE, 01SE로 8개 시그니처가 존재한다. 각 시그니처는 avi genericblock의 00dc, 01wb처럼 비디오 및 오디오 등의 데이터를 의미한다. 각 시그니처에 해당하는 구조체 데이터를 분석한 결과는 다음과 같다.

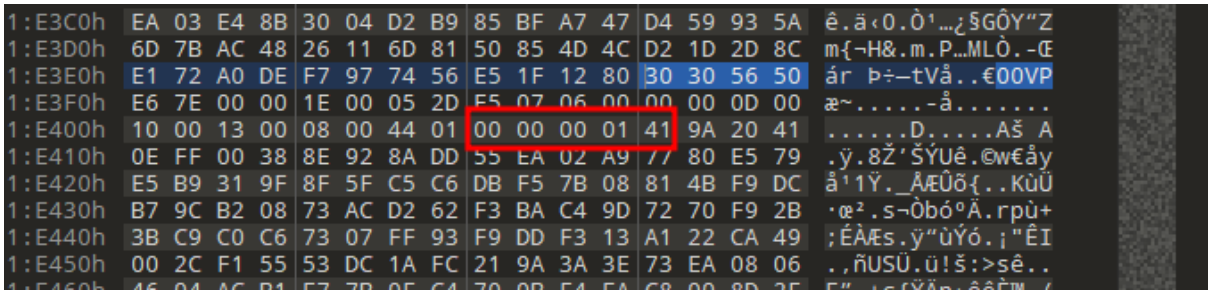
시그니처	의미
00VI	channel1 video i-frame data
01VI	channel2 video i-frame data
00VP	channel1 video p-frame data
01VP	channel2 video p-frame data
00AD	channel1 audio data
01AD	channel2 audio data
00SE	channel1 time data
01SE	channel2 time data

00VI, 01VI, 00VP, 01VP 비디오 프레임 구조체를 확인하면, 다음과 같이 시간 데이터가 저장되어 있는 것을 확인할 수 있다.



시간 데이터 이후에 00VI, 01VI는 i-frame 시그니처 '00 00 00 01 67'가 존재하고, 00VP, 01VP는 p-frame 시그니처 '00 00 00 01 41'가 존재한다.



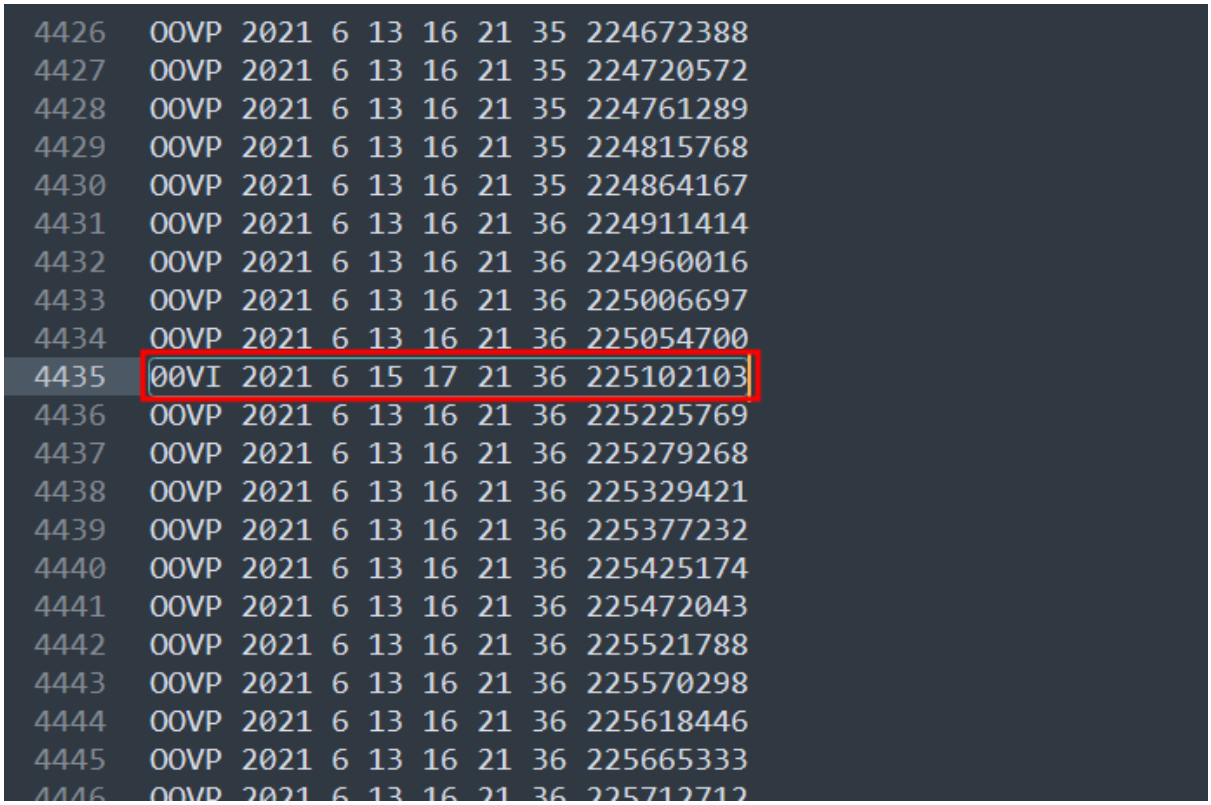


위에서 언급한 정보들을 기반으로 아래와 같이 python 코드를 이용하여, channel1의 비디오 데이터 추출 및 비디오 프레임별 시간 값들을 출력할 수 있다.

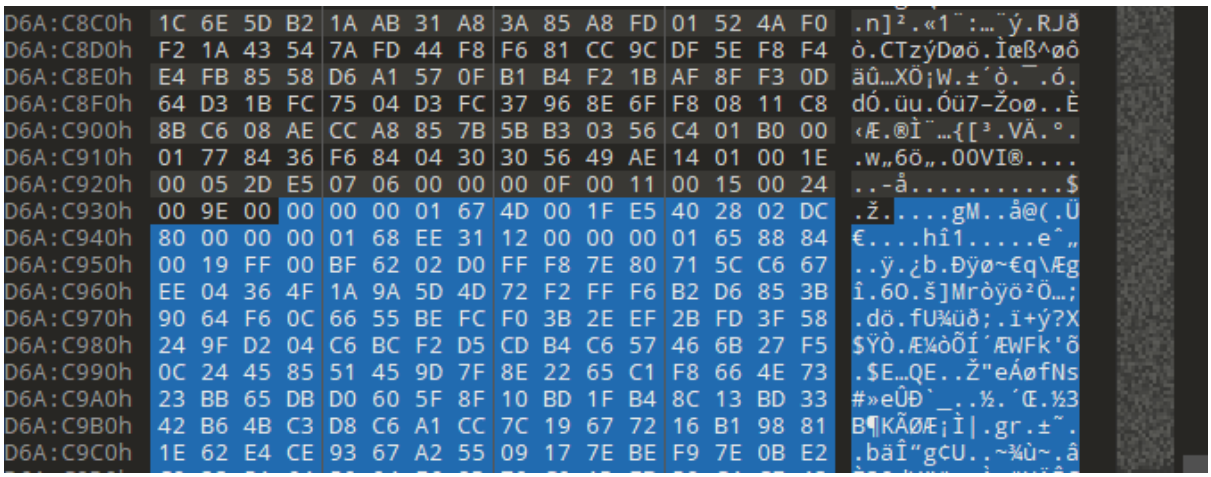
```
import struct

data = open('problem-final.img', 'rb').read()
video_ch1_fd = open('ch01.h264', 'wb')


def extract_VP(start_idx, end_idx):
    idx = start_idx
    while True:
        idx = data.find(bytes.fromhex('30 30 56 50'), idx + 1)  # 00VP
        if idx > end_idx or idx == -1:
            break
        if data[idx + 8] == 30 and data[idx + 12:idx + 14] !=
bytes.fromhex('30 30') and data[idx + 12:idx + 14] != bytes.fromhex('30
31'):
            size = struct.unpack('<I', data[idx + 4:idx + 8])[0]
            video_ch1_fd.write(data[idx + 28:idx + 28 + size])  # 00 00
00 01 41

            year = struct.unpack('<H', data[idx + 12:idx + 14])[0]
            month = struct.unpack('<H', data[idx + 14:idx + 16])[0]
            day = struct.unpack('<H', data[idx + 18:idx + 20])[0]
            h = struct.unpack('<H', data[idx + 20:idx + 22])[0]
            m = struct.unpack('<H', data[idx + 22:idx + 24])[0]
            s = struct.unpack('<H', data[idx + 24:idx + 26])[0]

            print('00VP', year, month, day, h, m, s, idx)


if __name__ == '__main__':
    idx = 0
    while True:
        idx = data.find(bytes.fromhex('30 30 56 49'), idx + 1) # 00VI
        if idx == -1:
            break
        elif data[idx + 8] == 30 and data[idx + 12:idx + 14] !=
bytes.fromhex('30 30') and data[idx + 12:idx + 14] != bytes.fromhex('30
31'):
            size = struct.unpack('<I', data[idx + 4:idx + 8])[0]
            video_ch1_fd.write(data[idx + 28:idx + 28 + size])  # 00 00
00 01 67

            year = struct.unpack('<H', data[idx + 12:idx + 14])[0]
            month = struct.unpack('<H', data[idx + 14:idx + 16])[0]
            day = struct.unpack('<H', data[idx + 18:idx + 20])[0]
            h = struct.unpack('<H', data[idx + 20:idx + 22])[0]
            m = struct.unpack('<H', data[idx + 22:idx + 24])[0]
            s = struct.unpack('<H', data[idx + 24:idx + 26])[0]

            print('00VI', year, month, day, h, m, s, idx)

            end_idx = data.find(bytes.fromhex('30 30 56 49'), idx + 1)
# 00VI
            if end_idx == -1:
```

```
                break
            else:
                extract_VP(idx, end_idx)
```

비디오 프레임별 시간 값들을 확인하면, 유일하게 오프셋 225102103에 위치한 프레임만 2021-06-15 일자인 것을 확인할 수 있다.



오프셋 225102103에 위치한 프레임 데이터만 추출하여, ffmpeg로 프레임에 해당하는 이미지를 추출한다.



```
root@Mal4ensics:/mnt/d/Workspace/Development/Python/ctf/Contest/2023_DFC/152/solve#
ffmpeg -i frame_225102103 result_%d.png
ffmpeg version 4.2.7-0ubuntu0.1 Copyright (c) 2000-2022 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.4.0-1ubuntu1~20.04.1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.1 --toolchain=hardened --lib
dir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --
enable-gpl --disable-stripping --enable-avresample --disable-filter=resample --enabl
e-avisynth --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-
libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --en
```

이미지를 확인해보면, 영상에 존재하는 시간 값(2021-06-13)과 프레임 데이터에 존재하는 시간 값(2021-06-15)이 다른 것을 확인할 수 있다.

*** Q1 Answer : Acoustic Analysis of Speech second edition**

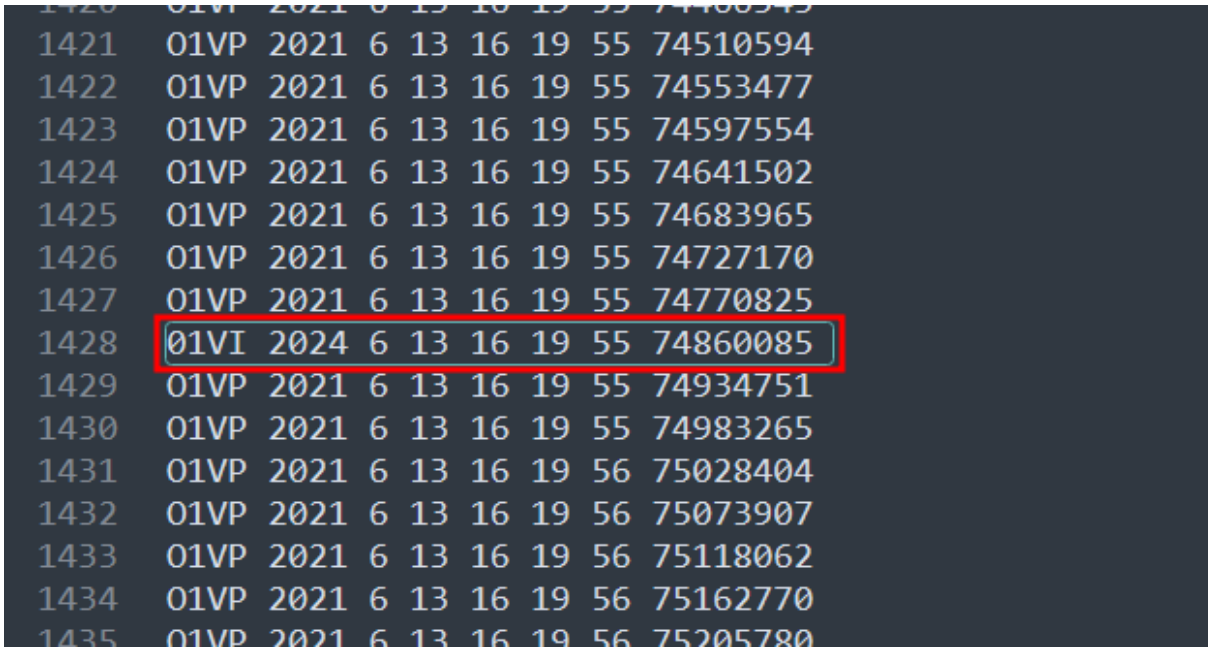
Q2. Submit the title of the book containing recorded video time with time information from the year 2024 (The time information of the video is located at the beginning of the video data). (100 points)

Q1 풀이에 언급된 정보들을 기반으로 아래와 같이 python 코드를 이용하여, channel2의 비디오 데이터 추출 및 비디오 프레임별 시간 값들을 출력할 수 있다.

```
import struct

data = open('problem-final.img', 'rb').read()
video_ch1_fd = open('ch02.h264', 'wb')


def extract_VP(start_idx, end_idx):
    idx = start_idx
    while True:
        idx = data.find(bytes.fromhex('30 31 56 50'), idx + 1)  # 01VP
        if idx > end_idx or idx == -1:
            break
        if data[idx + 8] == 30 and data[idx + 12:idx + 14] !=
bytes.fromhex('30 30') and data[idx + 12:idx + 14] != bytes.fromhex('30
31'):
            size = struct.unpack('<I', data[idx + 4:idx + 8])[0]
            video_ch1_fd.write(data[idx + 28:idx + 28 + size])  # 00 00
00 01 41

            year = struct.unpack('<H', data[idx + 12:idx + 14])[0]
            month = struct.unpack('<H', data[idx + 14:idx + 16])[0]
            day = struct.unpack('<H', data[idx + 18:idx + 20])[0]
            h = struct.unpack('<H', data[idx + 20:idx + 22])[0]
            m = struct.unpack('<H', data[idx + 22:idx + 24])[0]
            s = struct.unpack('<H', data[idx + 24:idx + 26])[0]

            print('01VP', year, month, day, h, m, s, idx)


if __name__ == '__main__':
    idx = 0
    while True:
        idx = data.find(bytes.fromhex('30 31 56 49'), idx + 1) # 01VI
        if idx == -1:
            break
        elif data[idx + 8] == 30 and data[idx + 12:idx + 14] !=
bytes.fromhex('30 30') and data[idx + 12:idx + 14] != bytes.fromhex('30
31'):
            size = struct.unpack('<I', data[idx + 4:idx + 8])[0]
            video_ch1_fd.write(data[idx + 28:idx + 28 + size])  # 00 00
00 01 67

            year = struct.unpack('<H', data[idx + 12:idx + 14])[0]
            month = struct.unpack('<H', data[idx + 14:idx + 16])[0]
            day = struct.unpack('<H', data[idx + 18:idx + 20])[0]
            h = struct.unpack('<H', data[idx + 20:idx + 22])[0]
            m = struct.unpack('<H', data[idx + 22:idx + 24])[0]
            s = struct.unpack('<H', data[idx + 24:idx + 26])[0]
```

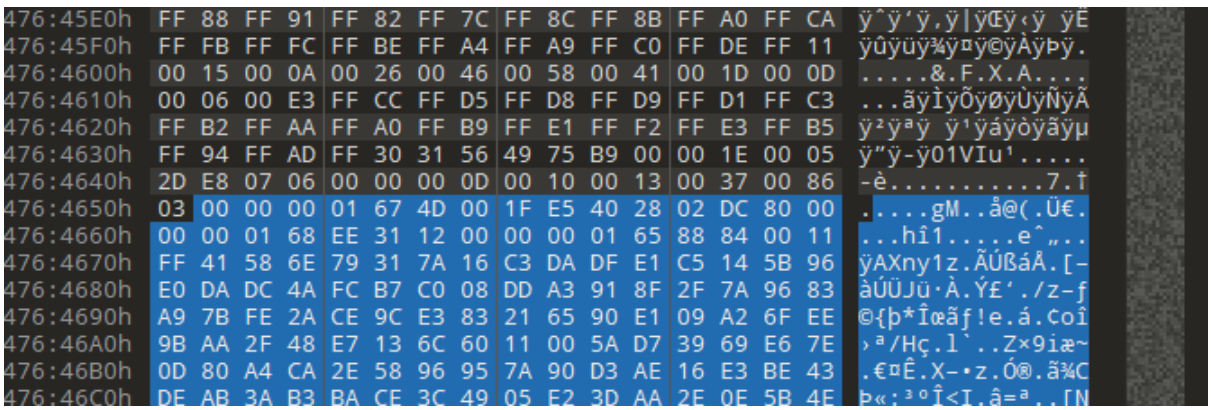
```
            print('01VI', year, month, day, h, m, s, idx)

            end_idx = data.find(bytes.fromhex('30 31 56 49'), idx + 1)
# 01VI
            if end_idx == -1:
                extract_VP(idx, len(data) - 1)
                break
            else:
                extract_VP(idx, end_idx)
```

비디오 프레임별 시간 값들을 확인하면, 유일하게 오프셋 74860085에 위치한 프레임만 2024년 이후 일자인 것을 확인할 수 있다.



1421	O1VP	2021	6	13	16	19	55	74510594
1422	O1VP	2021	6	13	16	19	55	74553477
1423	O1VP	2021	6	13	16	19	55	74597554
1424	O1VP	2021	6	13	16	19	55	74641502
1425	O1VP	2021	6	13	16	19	55	74683965
1426	O1VP	2021	6	13	16	19	55	74727170
1427	O1VP	2021	6	13	16	19	55	74770825
1428	01VI	2024	6	13	16	19	55	74860085
1429	O1VP	2021	6	13	16	19	55	74934751
1430	O1VP	2021	6	13	16	19	55	74983265
1431	O1VP	2021	6	13	16	19	56	75028404
1432	O1VP	2021	6	13	16	19	56	75073907
1433	O1VP	2021	6	13	16	19	56	75118062
1434	O1VP	2021	6	13	16	19	56	75162770

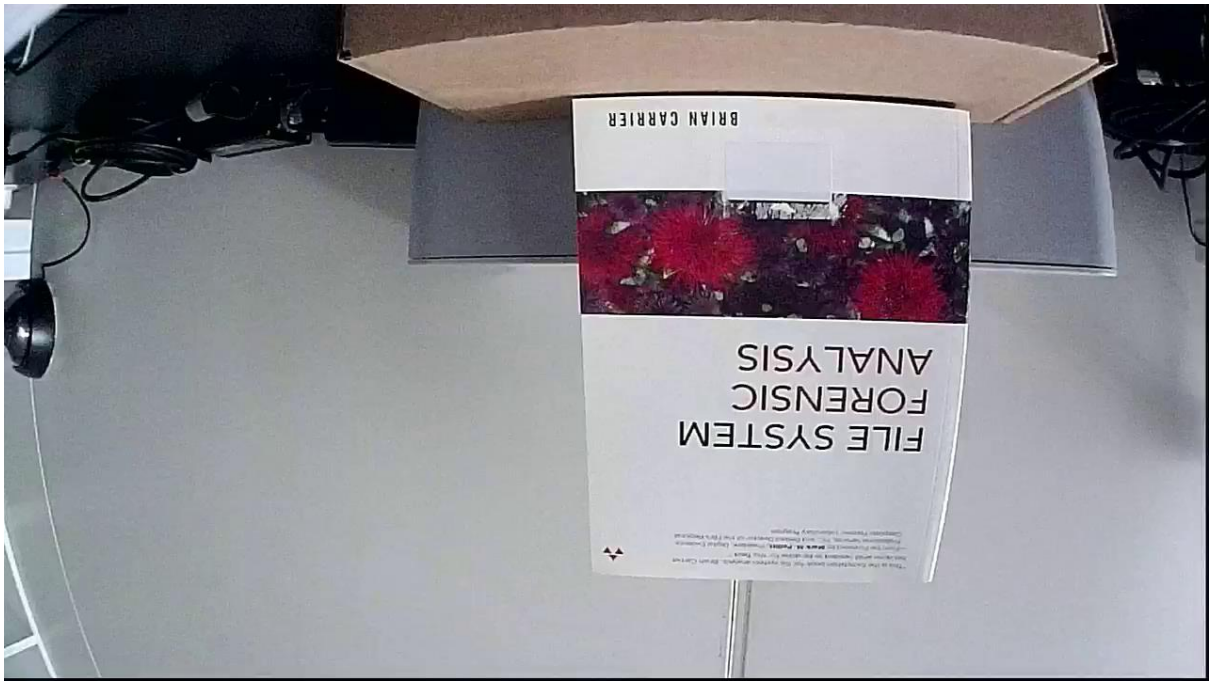
오프셋 74860085에 위치한 프레임 데이터만 추출하여, ffmpeg로 프레임에 해당하는 이미지를 추출한다.



```
476:45E0h  FF 88 FF 91 FF 82 FF 7C FF 8C FF 8B FF A0 FF CA  ÿˆÿ'ÿ‚ÿ|ÿŒÿ‹ÿ ÿÊ
476:45F0h  FF FB FF FC FF BE FF A4 FF A9 FF C0 FF DE FF 11  ÿûÿüÿ¾ÿ¤ÿ©ÿÀÿÞÿ.
476:4600h  00 15 00 0A 00 26 00 46 00 58 00 41 00 1D 00 0D  .....&.F.X.A....
476:4610h  00 06 00 E3 FF CC FF D5 FF D8 FF D9 FF D1 FF C3  ...ãÿÌÿÕÿØÿÙÿÑÿÃ
476:4620h  FF B2 FF AA FF A0 FF B9 FF E1 FF F2 FF E3 FF B5  ÿ²ÿªÿ ÿ¹ÿáÿòÿãÿµ
476:4630h  FF 94 FF AD FF 30 31 56 49 75 B9 00 00 1E 00 05  ÿ"ÿ-ÿ01VIu¹.....
476:4640h  2D E8 07 06 00 00 00 0D 00 10 00 13 00 37 00 86  -è...........7.†
476:4650h  03 00 00 00 01 67 4D 00 1F E5 40 28 02 DC 80 00  .....gM..å@(.Ü€.
476:4660h  00 00 01 68 EE 31 12 00 00 00 01 65 88 84 00 11  ...hî1.....eˆ„..
476:4670h  FF 41 58 6E 79 31 7A 16 C3 DA DF E1 C5 14 5B 96  ÿAXny1z.ÃÚßáÅ.[-
476:4680h  E0 DA DC 4A FC B7 C0 08 DD A3 91 8F 2F 7A 96 83  àÚÜJü·À.Ý£'./z-ƒ
476:4690h  A9 7B FE 2A CE 9C E3 83 21 65 90 E1 09 A2 6F EE  ©{þ*Îœãƒ!e.á.¢oî
476:46A0h  9B AA 2F 48 E7 13 6C 60 11 00 5A D7 39 69 E6 7E  ›ª/Hç.l`..Z×9iæ~
476:46B0h  0D 80 A4 CA 2E 58 96 95 7A 90 D3 AE 16 E3 BE 43  .€¤Ê.X-•z.Ó®.ã¾C
476:46C0h  DE AB 3A B3 BA CE 3C 49 05 E2 3D AA 2E 0E 5B 4E  Þ«:³ºÎ<I.â=ª..[N
```

```
root@Mal4ensics:/mnt/d/Workspace/Development/Python/ctf/Contest/2023_DFC/152/solve#
ffmpeg -i frame_74860085 result_%d.png
ffmpeg version 4.2.7-0ubuntu0.1 Copyright (c) 2000-2022 the FFmpeg developers
  built with gcc 9 (Ubuntu 9.4.0-1ubuntu1~20.04.1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.1 --toolchain=hardened --lib
dir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --
enable-gpl --disable-stripping --enable-avresample --disable-filter=resample --enabl
e-avisynth --enable-gnutls --enable-ladspa --enable-libaom --enable-libass --enable-
libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --en
able-libflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enab
le-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --
```

이미지를 확인해보면, 'File System Forensic Analysis' 책을 확인할 수 있다.



*** Q2 Answer : File System Forensic Analysis**