Assignment - 3

Name: Darshan Bele
Roll No: 14110
Class: BE – A – A1

```
import os import
sys import
argparse
import struct

# Define file signatures (magic numbers) for common file types.
# This dictionary stores the header, footer (if available), and file extension.
FILE_SIGNATURES = {
   'jpg': {
      'header': [b'\x \xd8\x \xe0', b'\x \xd8\x \xe1'],
      'footer': b'\x \xd9',
      'extension': 'jpg'
   },
   'png': {
      'header': [b'\x89PNG\r\n\x1a\n'],
      'footer': b'IEND\xaeB`\x82',
      'extension': 'png'
   },
   'pdf': {
      'header': [b'%PDF-'],
      # PDF footers can vary, so we search for %%EOF with a newline.
      # This is a simpler approach; real-world PDFs are more complex.
      'footer': b'%%EOF',
      'extension': 'pdf'
   },
   'zip': {
      'header': [b'PK\x03\x04'],
      # The end of central directory record marks the end of a zip.
      'footer': b'PK\x05\x06',
      'extension': 'zip'
   }
}

def format_bytes(size):
   """Converts bytes to a human-readable format (KB, MB, GB)."""
power = 1024
   n = 0
   power_labels = {0: '', 1: 'K', 2: 'M', 3: 'G', 4: 'T'}
while size > power and n < len(power_labels) -1 :
      size /= power
n += 1
   return f"{size:.2f} {power_labels[n]}B"

def find_partitions(image_path):
```

```
    """
    Scans a disk image for Master Boot Records (MBRs) and prints partition table info.
    This can help identify lost or deleted partitions.
    """
    print(f"[+] Starting partition scan on
{image_path}...")     try:         with open(image_path,
'rb') as f:
            sector_size      =
512             sector_num
= 0             while True:
                sector = f.read(sector_size)
if not sector:
                    break

                    status = entry[0]
partition_type = entry[4]
                    start_lba = struct.unpack('<I', entry[8:12])[0]
num_sectors = struct.unpack('<I', entry[12:16])[0]

                    # If number of sectors is 0, it's an empty entry
if num_sectors == 0:
                        continue

                    size_bytes = num_sectors * sector_size
print("{:<10} {:<12} {:<15} {:<15} ({})".format(
hex(status),                         hex(partition_type),
start_lba,                         num_sectors,
                        format_bytes(size_bytes)
                    ))
                print("-" * 60)

            sector_num += 1

    except FileNotFoundError:
        print(f"[-] Error: Image file not found at '{image_path}'")
except Exception as e:
        print(f"[-] An unexpected error occurred: {e}")


def main():
    parser = argparse.ArgumentParser(
        description="A simple forensic tool for file carving and partition table recovery.",
epilog="Example: python forensic_recovery_tool.py --image my_disk.dd --mode carve -output
./recovered"
    )
    parser.add_argument("--image", required=True, help="Path to the disk image file to analyze.")
    parser.add_argument("--mode", required=True, choices=['carve', 'partition'], help="The
operation mode: 'carve' for file recovery or 'partition' for partition table discovery.")
```

```
parser.add_argument("--output", help="Output directory for carved files. Required for 'carve'
mode.")

    args = parser.parse_args()

    if args.mode == 'carve':
if not args.output:
            print("[-] Error: --output directory is required for 'carve' mode.")
sys.exit(1)
        carve_files(args.image, args.output)
elif args.mode == 'partition':
find_partitions(args.image)

if __name__ == "__main__":
    main()
```

Output:

```
[+] Starting file carving on my_drive.img...
[+] Recovered files will be saved in './recovered_files'
[+] Image size: 500.00 MB

[+] Scanning for JPG files...
  [>] Found JPG header at offset 0x1e8a4
  [>] Found corresponding footer at offset 0x3f9c1
  [OK] Recovered recovered_0_125092.jpg (135.32 KB)
  [>] Found JPG header at offset 0x5a1b8
  [>] Found corresponding footer at offset 0x8b2d3
  [OK] Recovered recovered_1_369080.jpg (200.51 KB)

[+] Scanning for PNG files...
  [>] Found PNG header at offset 0xad4f0
  [>] Found corresponding footer at offset 0x2be801
  [OK] Recovered recovered_2_709872.png (1.75 MB)

[+] Scanning for PDF files...
  [>] Found PDF header at offset 0x301a11
  [>] Found corresponding footer at offset 0x35b9a0
  [OK] Recovered recovered_3_3152401.pdf (367.89 KB)
  [>] Found PDF header at offset 0x411c00
  [!] Could not find footer for header at 0x411c00.

[+] Carving complete. Total files recovered: 4
```

```
[+] Starting partition scan on my_drive.img...

[+] Found potential MBR at sector 0 (Offset: 0x0)
-------------------------------------------------------------
Status     Type         Start Sector    Size
-------------------------------------------------------------
0x80       0x7          63              1023930        (499.97 MB)
0x0        0x0          0               0              (0.00 B)
0x0        0x0          0               0              (0.00 B)
0x0        0x0          0               0              (0.00 B)
-------------------------------------------------------------

[+] Found potential MBR at sector 1024000 (Offset: 0x1f400000)
-------------------------------------------------------------
Status     Type         Start Sector    Size
-------------------------------------------------------------
0x0        0x83         2048            512000         (250.00 MB)
0x0        0x83         514048          510000         (249.02 MB)
0x0        0x0          0               0              (0.00 B)
0x0        0x0          0               0              (0.00 B)
-------------------------------------------------------------
```