❖ **Assignment No:** 3

❖ **Problem Statement:** Write a computer forensic application program for Recovering Permanent deleted files and Deleted Partitions

❖ **Objectives:**

- To develop a computer forensic tool for recovering permanently deleted files and lost partitions.
- To understand and implement basic file carving and partition recovery techniques.
- To assist investigators in restoring evidence from damaged or formatted storage media.

❖ **Hardware Requirements:** Intel Core i3 or higher Processor, 4 GB RAM, 250 GB Hard Disk, USB drive or hard disk (for recovery testing)

❖ **Software Requirements:** Windows 10 / Linux / macOS Operating System, PyCharm, VS Code, or any Python IDE

❖ **Theory:**

- **Computer Forensics and Data Recovery**

  Computer forensics is the process of identifying, preserving, analyzing, and presenting digital evidence in a legally acceptable manner. One of its major applications is **data recovery**, which involves retrieving deleted or lost data from storage devices like hard drives, pen drives, or memory cards.

i. **File Deletion and Recovery**

When a file is "deleted," it is not immediately removed from the disk. The operating system only marks the file's storage space as *available* for new data. Until new data overwrites this space, the file's content remains recoverable.

File recovery tools use **file carving techniques**, which identify file headers and footers (unique byte patterns) to reconstruct deleted files without relying on file system metadata.

ii. Partition Loss and Recovery

A partition may be lost due to formatting, virus attacks, or corruption of the Master Boot Record (MBR). Recovery involves scanning the disk for partition signatures and rebuilding the partition table.

iii. Common Recovery Techniques

- **File Signature Analysis:** Identifies deleted files by their unique header/footer bytes (e.g., JPG, PNG, PDF).
- **Data Carving:** Extracts raw data blocks and reconstructs files.
- **Partition Table Analysis:** Restores deleted or corrupted partitions using disk structure information.

iv. Forensic Importance

In cybercrime investigations, recovering deleted data helps identify evidence such as documents, images, or logs that may have been intentionally removed. Forensic recovery tools ensure data integrity and maintain a clear chain of custody.

v. Applications

- Digital crime investigation
- Accidental data recovery
- File system repair and analysis
- Evidence restoration for legal cases

❖ **Conclusion:**

This practical demonstrates the process of recovering permanently deleted files and lost partitions using forensic techniques. It shows how data carving and partition scanning can restore lost digital evidence, aiding cybersecurity experts and forensic investigators in analyzing storage devices effectively.

❖ **Code:**

```
import os
from pathlib import Path
from PIL import Image
from IPython.display import display
import requests

# Step 1: Setup paths
workdir = Path("forensic_recovery_demo")
disk_image_path = workdir / "disk_image.bin"
recovered_image_path = workdir / "recovered_image.jpg"
workdir.mkdir(parents=True, exist_ok=True)

# Step 2: Download JPG image from web
url = "https://images.pexels.com/photos/674010/pexels-photo-674010.jpeg?cs=srgb&dl=pexels-anjana-c-169994-674010.jpg&fm=jpg"
original_image_path = workdir / "pexels-anjana-c-169994-674010.jpg"
r = requests.get(url)
with open(original_image_path, "wb") as f:
    f.write(r.content)

# Step 3: Create simulated disk image with JPG
with open(disk_image_path, "wb") as f:
    f.write(b"Random data before image...\n")
    with open(original_image_path, "rb") as img_file:
        f.write(img_file.read())
    f.write(b"\nRandom data after image...")

print(f"Disk image created at: {disk_image_path}")

# Step 4: Recover JPG from disk image
def recover_jpg(disk_image_path, output_path):
    with open(disk_image_path, "rb") as f:
        data = f.read()
```

```
    start = data.find(b'\xff\xd8')  # JPG start
    end = data.find(b'\xff\xd9') + 2  # JPG end
    if start != -1 and end != -1:
        jpg_bytes = data[start:end]
        with open(output_path, "wb") as out_file:
            out_file.write(jpg_bytes)
        print("✅ JPG recovered successfully!")
        return output_path
    else:
        print("❌ JPG not found in disk image.")
        return None

# Step 5: Recover and display the JPG
recovered_file = recover_jpg(disk_image_path, recovered_image_path)
if recovered_file:
    display(Image.open(recovered_file))
```

❖ **Output:**

```
Disk image created at: forensic_recovery_demo\disk_image.bin
✅ JPG recovered successfully!
```