Ex. No.: 18

Date:

FILE ORGANISATION TECHNIQUES

a. SINGLE LEVEL:

CODE:

```
dir = {
    'dname': '',
    'files':{}
}

dir['fcnt'] = 0
dir['dname'] = input("Enter name of directory -- ")

while True:
    print("\n\n 1. Create File\t2. Delete File\t3. Search File \n 4. Display
Files\t5. Display file content\t6. Exit")
    ch = int(input("Enter your choice -- "))

    if ch == 1:
        fname = input("\n Enter the name of the file -- ")
        content=input("Enter file contents --")
        dir['files'][fname]=content

    elif ch == 2:
        f = input("\n Enter the name of the file -- ")
        if f in dir['files']:
            del dir['files'][f]
            print("File",f,"found and deleted")
        else:
            print("File", f, "not found")
            dir['fcnt'] -= 1

    elif ch == 3:
        f = input("\n Enter the name of the file -- ")
        if f in dir['files']:
            print("File",f,"found")
        else:
            print("File", f, "not found")

    elif ch == 4:
        if len(dir['files'])==0:
            print("\n Directory Empty")
        else:
            print("\n The Files are -- ")
```

```python
            for i in dir['files']:
                print(i,end="\n")

    elif ch == 5:
        f = input("\n Enter the name of the file -- ")
        if f in dir['files']:
            print("Content:")
            print(dir['files'][f])
        else:
            print("File not found")
    else:
        print("Invalid option!!")
        break
```

OUTPUT:

```
Enter name of directory -- D1


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 1

 Enter the name of the file -- F1
Enter file contents --HI


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 1

 Enter the name of the file -- F2
Enter file contents --HELLO


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 1

 Enter the name of the file -- F3
Enter file contents --BRO


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 2

 Enter the name of the file -- F3
File F3 found and deleted
```

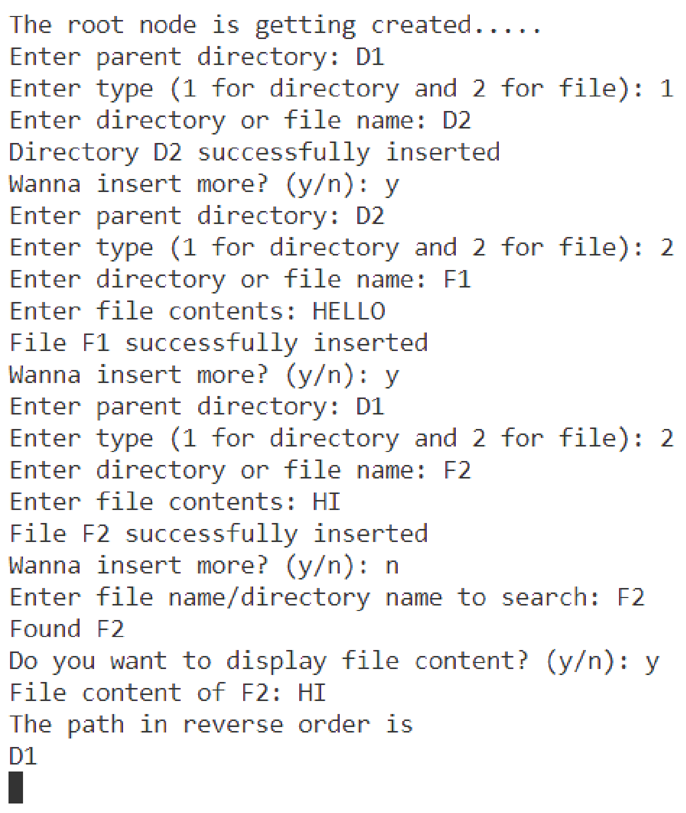
```
 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 3

 Enter the name of the file -- F3
File F3 not found


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 3

 Enter the name of the file -- F2
File F2 found


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 4

 The Files are --
F1
F2


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 5

 Enter the name of the file -- F2
Content:
HELLO


 1. Create File 2. Delete File  3. Search File
 4. Display Files       5. Display file content 6. Exit
Enter your choice -- 6
Invalid option!!
```

c. TREE LEVEL:

CODE:

```
class Node:
    def __init__(self, name, type):
        self.name = name
        self.type = type
        self.next = None
        self.down = None
        self.content = None

def new_node(item, type1, content=None):
    temp = Node(item, type1)
    temp.next = None
    temp.down = None
    temp.content = content
    return temp


def inorder(root, p):
    if root.next != None and root.name != p:
        inorder(root.next, p)
        print(root.name)
        if root.type == 1:
            inorder(root.down, p)
    return root

def find(node, key):
    if node is not None:
        if node.name == key:
            print(f"Found {node.name}")
            return node
        else:
            found_node = find(node.down, key)
            if found_node is None:
                found_node = find(node.next, key)
            return found_node
    return None


def insert(node, key, par, mode, content=None):
    if node is None:
        print("The root node is getting created.....")
        return new_node(key, mode, content)
    else:
        temp = None
        temp = inorder(node, par)
        temp1 = new_node(key, mode, content)  # Pass content parameter here
        if temp.down is None and temp.type == 1:
            temp.down = temp1
            if temp1.type == 2:
                print(f"File {temp1.name} successfully inserted")
            else:
                print(f"Directory {temp1.name} successfully inserted")
        else:
            temp = temp.down
            while temp.next is not None:
                temp = temp.next
            temp.next = temp1
            if temp1.type == 2:
                print(f"File {temp1.name} successfully inserted")
            else:
                print(f"Directory {temp1.name} successfully inserted")
    return node

root = None
c = 0
p = 0
parent = [None] * 50
child = [None] * 50
cont = 'y'
root = insert(root, "root", "", 1)
while cont == 'y':
    par_dir = input("Enter parent directory: ")
    t = int(input("Enter type (1 for directory and 2 for file): "))
    file_or_dir = input("Enter directory or file name: ")


    # Ask for file contents if it's a file
    content = None
    if t == 2:
        content = input("Enter file contents: ")

    insert(root, file_or_dir, par_dir, t, content)

    child[c] = file_or_dir
    parent[p] = par_dir
    c += 1
    p += 1
    cont = input("Wanna insert more? (y/n): ")

finder = input("Enter file name/directory name to search: ")
found_node = find(root, finder)

if found_node:
    option = input("Do you want to display file content? (y/n): ")
    if found_node.content==None:
        print("It's a directory")

    elif option.lower() == 'y' and found_node.type == 2 and 
found_node.content:
        print(f"File content of {found_node.name}: {found_node.content}")

        par = ""
        chi = found_node.name
        print("The path in reverse order is")
        while par != "root":
            for i in range(c):
                if child[i] == chi:
                    par = parent[i]
                    chi = parent[i]
                    print(par)
                    break
```

OUTPUT:



```
The root node is getting created.....
Enter parent directory: D1
Enter type (1 for directory and 2 for file): 1
Enter directory or file name: D2
Directory D2 successfully inserted
Wanna insert more? (y/n): y
Enter parent directory: D2
Enter type (1 for directory and 2 for file): 2
Enter directory or file name: F1
Enter file contents: HELLO
File F1 successfully inserted
Wanna insert more? (y/n): y
Enter parent directory: D1
Enter type (1 for directory and 2 for file): 2
Enter directory or file name: F2
Enter file contents: HI
File F2 successfully inserted
Wanna insert more? (y/n): n
Enter file name/directory name to search: F2
Found F2
Do you want to display file content? (y/n): y
File content of F2: HI
The path in reverse order is
D1
```

B. TWO LEVEL

```
import os
class File:
    def __init__(self, filename, content):
        self.filename = filename
        self.content = content
class UserDirectory:
    def __init__(self, path):
        self.path = path
        self.files = {}

def create_user_directory(system, master_directory, username):
 # Create a new user directory for the given username.
    if username not in system.master_directory:
        user_path = os.path.join(master_directory, username)
        os.makedirs(user_path, exist_ok=True)
        system.master_directory[username] = UserDirectory(user_path)

def create_new_file_in_user_directory(system, username, filename, content):
 # Create a new file with content in the user's directory.
    if username in system.master_directory:
        user_directory = system.master_directory[username]
        file_path = os.path.join(user_directory.path, filename)
        with open(file_path, 'w') as file:
            file.write(content)
        user_directory.files[filename] = File(filename, content)
    else:
        print(f"User '{username}' not found. Create the user directory first.")

def list_user_files(system, username):
 # List the files in the user's directory.
    if username in system.master_directory:
        user_directory = system.master_directory[username]
        print(f"Files in User '{username}' Directory:")
        for filename in user_directory.files:
            print(filename)

def read_file(system, username, filename):
 # Read the content of a file in the user's directory.
    if username in system.master_directory:
        user_directory = system.master_directory[username]
        if filename in user_directory.files:
            file = user_directory.files[filename]
            return file.content
        else:
            return f"File '{filename}' not found in User '{username}' Directory."
    else:
        return f"User '{username}' not found."
```

```
# Interactive driver code
class TwoLevelDirectorySystem:
    def __init__(self):
 # The master directory, a dictionary with usernames as keys and user directories
as values.
        self.master_directory = {}

master_directory_path = input("Enter the path where the master directory should be
created: ")
if not os.path.exists(master_directory_path):
     os.makedirs(master_directory_path)

system = TwoLevelDirectorySystem()
while True:
    print("\nOptions:")
    print("1. Create User Directory")
    print("2. Create New File in User Directory")
    print("3. List User Files")
    print("4. Read File Content")
    print("5. Exit")
    choice = input("Enter your choice: ")
    if choice == '1':
        username = input("Enter the username: ")
        create_user_directory(system, master_directory_path, username)
    elif choice == '2':
        username = input("Enter the username: ")
        filename = input("Enter the filename: ")
        content = input("Enter the content for the new file: ")
        create_new_file_in_user_directory(system, username, filename, content)
    elif choice == '3':
        username = input("Enter the username: ")
        list_user_files(system, username)
    elif choice == '4':
        username = input("Enter the username: ")
        filename = input("Enter the filename: ")
        content = read_file(system, username, filename)
        print(content)
    elif choice == '5':
        break
    else:
        print("Invalid choice. Please enter a valid option.")
```

D. ACYCLIC GRAPH

```python
class File:
    def __init__(self, path):
        self.path = path

class Directory:
    def __init__(self, dname):
        self.dname = dname
        self.files = []

def create_directory(path, dname, files):
    directory_path = os.path.join(path, dname)
    os.makedirs(directory_path, exist_ok=True)

    directory = Directory(directory_path)
    for file in files:
        directory.files.append(file)
    return directory

def search_file(fname):
    matches = []
    for directory in directories:
        for file in directory.files:
            if fname in file.path:
                matches.append((directory.dname, file.path))

    if matches:
        print("\nMatch(es) found:")
        for directory_name, match in matches:
            print(f"In Directory '{directory_name}': {match}")

count = int(input("Enter the number of base directories: "))
directories = []
for _ in range(count):
    base_path = input("Enter the base directory path: ")
    dname = input("Enter the directory name: ")
    fcount = int(input("Enter the number of files in the directory: "))
    files = []
    for _ in range(fcount):
        path = input("Enter file path: ")
        files.append(File(path))
    directories.append(create_directory(base_path, dname, files))
search_key = input("Enter the file to search: ")
search_file(search_key)
```