DATA STRUCTURES AND ITS APPLICATIONS (UE22CS252A)

Mini Project

Project Title & Team Members

Title

File Management System

Team Members (SRN Name)

- Nannapaneni Sashank (PES2UG22CS338)
- Nikhil Upadhyay (PES2UG22CS358)
- Nishank Bhowal (PES2UG22CS366)
- Nishanth J Modpur (PES2UG22CS369)

Introduction

The file management system is designed to simplify the management of files for users. It offers a straightforward and intuitive interface, presenting a range of options to manage files efficiently by providing the following functionalities:

1. Creating a new file
2. Deleting a file
3. Displaying all the files in the directory
4. Searching for a particular file
5. Restoring deleted files
6. Viewing all files with a particular file extension

Data Structures Used

1. Linked List
2. Binary Search Tree

ADT Definition of Data Structures Used

- SLOT *createFile(char fname[]):
  Creates a new file node with the given name and inserts it into the directory tree.
- SLOT *insertFile(SLOT *root, char fname[]):
  Inserts a new file node into the directory tree while maintaining the binary search tree property.
- SLOT *deleteFile(SLOT *root, char fname[]):
  Deletes a file node with the given name from the directory tree.
- void inOrderTraversal(SLOT *root):
  Performs an in-order traversal of the directory tree to display all files and directories.
- SLOT *searchFile(SLOT *root, char fname[]):
  Searches for a file with the given name in the directory tree.
- void displayFilesWithExtension(SLOT *root, char extension[]): Displays files with a specific extension in the directory tree.

ADT Definition of Data Structures Used

- SLOT *leftmostnode(SLOT *node):
  Finds the leftmost node in the directory tree.
  (Returns the least value node on the right of the current node.)
- LIST *createList(char fname[]):
  Creates a new node to store the name of a deleted file and inserts it into the list.
- LIST *insertToList(LIST *head, char fname[]):
  Inserts a new node at the end of the list to record a deleted file.
- LIST *restoreDeletedFile(LIST *head, char fname[]):
  Searches for a deleted file in the list and returns it if found, allowing restoration.

Creating File

- User enters the name of the file to be created, which calls a function ‘insertFile’ with parameters as root of the tree and the file name.
- if the filename to be inserted is lexicographically less than the current node's filename, it means that the new file should be inserted into the left subtree of the current node. It recursively calls insertFile on the left subtree, passing the current node's left child as the new root.
- Similarly, insertFile is called on the right subtree, passing the current node's right child as the new root if the filename to be inserted is lexicographically greater than the current node's filename.
- If (root == Null), it means that the BST is either empty or the function has reached a leaf node. Here, another function ‘createFile’ is called, which is used to initialise a node with the name of the file and print that the file was created.
- Otherwise, if the file name entered already exists, the user is notified about the same.

Deleting File

- First, a head pointer is made use of to store the files to be deleted in a linked list if the files exists, so that the linked list can be made use of to recover deleted files in the future.
- Then, the deleteFile function is called with the parameters as root and file name.
- If the filename to be deleted is lexicographically less than the current node's filename, it recursively calls deleteFile on the left subtree, and if the filename to be deleted is lexicographically greater than the current node's filename, it recursively calls deleteFile on the right subtree.
- If the file has no left child, the file is freed and the right child is returned.
- Similarly, if the file has no right child, the file is freed and the left child is returned.
- If the file has both left and right child, the left-most file in the right sub-tree (inorder successor) is found, its name is copied to the root file, and recursively calls deleteFile on the right subtree to delete the leftmost node.

Searching for a File

- The file name and root pointer is passed to a function called ‘searchFile’.
- While the root pointer isn’t NULL, the result of the lexicographical comparison between the file name to be deleted and the root’s file name is stored in a variable called ‘findFile’.
- If ‘findFile’ equals 0, it means that the file names are exactly equal and that the file has been found.
- If ‘findFile’ is less than 0, the root pointer is updated to its left child to search the left subtree.
- If ‘findFile’ is greater than 0, the root pointer is updated to its right child to search the right subtree.
- If root equals NULL, it either means that the binary tree is empty, or we have reached to the end of the tree without finding the desired file and the user is notified about the same, and NULL is returned.

Displaying Files in Current Directory

- To display the files in the current directory, the inorder traversal is made use of by passing the pointer to root node as the argument.
- If pointer to root node doesn’t point to NULL, the function ‘inOrderTraversal’ is called recursively first for the left subtree.
- Then the file name of the root node is printed.
- Then the function is called recursively for the right subtree.
- This ensures that the files are displayed in the order in which they were inserted into the the binary search tree.

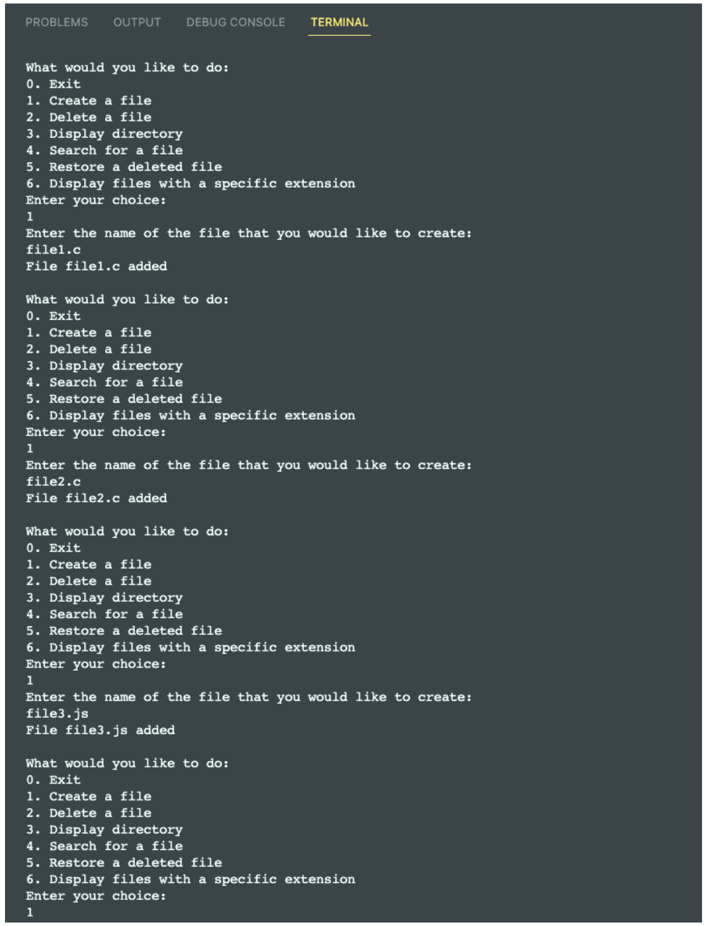
Restoring Deleted Files

- The user first enters the name of the file which is to be restored. This is passed as input a function ‘restoreDeletedFile’ along with the head pointer which was used to insert files into a linked list.
- If head equals NULL, it means that there is no file present to restore.
- Else, a current pointer is used to traverse the linked list. If the file name and the current pointer’s file name are the same, the current pointer is returned, else, current pointer updates to the next file in the list.
- If current pointer equals NULL, it means that the entire list was searched and the file to be restored was not found.
- The returned pointer from the function is stored in a pointer variable called ‘restore’. If ‘restore’ doesn’t equal NULL, the file pointed to by ‘restore’ pointer is inserted into the the Binary tree using ‘insertFile’ function.

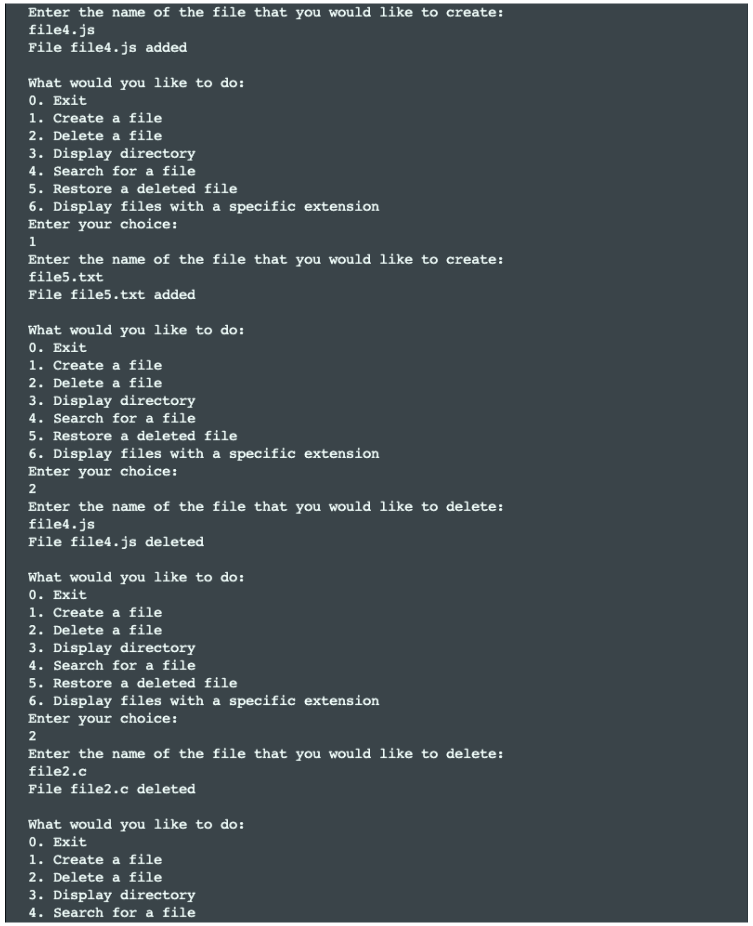
Viewing Files with Specific Extension

- The user passes as input the extension, after which a function ‘displayFilesWithExtension’ is called with parameters as the root pointer and the extension name.
- An array of pointers to SLOT structures is used, acting as a stack for iterative traversal.
- Variable ‘top’ is used to keep track of top of stack and variable ‘found’ is used to check if a file has been found or not.
- Nodes are pushed onto the stack until the leftmost node has been reached
- After reaching the leftmost node, it pops a node from the stack and extracts the file extension from the node's file name using the ‘strrchr’ function which finds the last occurrence of the ‘.’ (dot) character.
- If the extension passed as attribute matches the extension extracted from file name while traversing through the files, the file names are displayed as output and found is set to 1.
- The function then moves to the right subtree of the current node (root = root->right) to continue the in-order traversal.
- If ‘found’ equals 0, no files were found with the specified extension and the user is informed about the same.

Output Screenshots

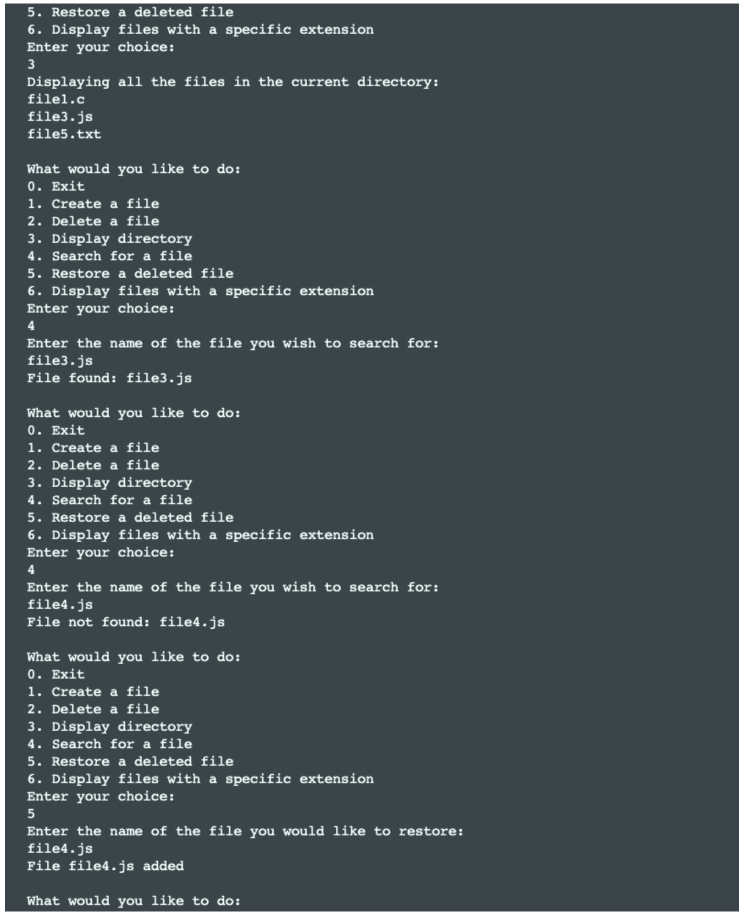

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
1
Enter the name of the file that you would like to create:
file1.c
File file1.c added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
1
Enter the name of the file that you would like to create:
file2.c
File file2.c added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
1
Enter the name of the file that you would like to create:
file3.js
File file3.js added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
1
```

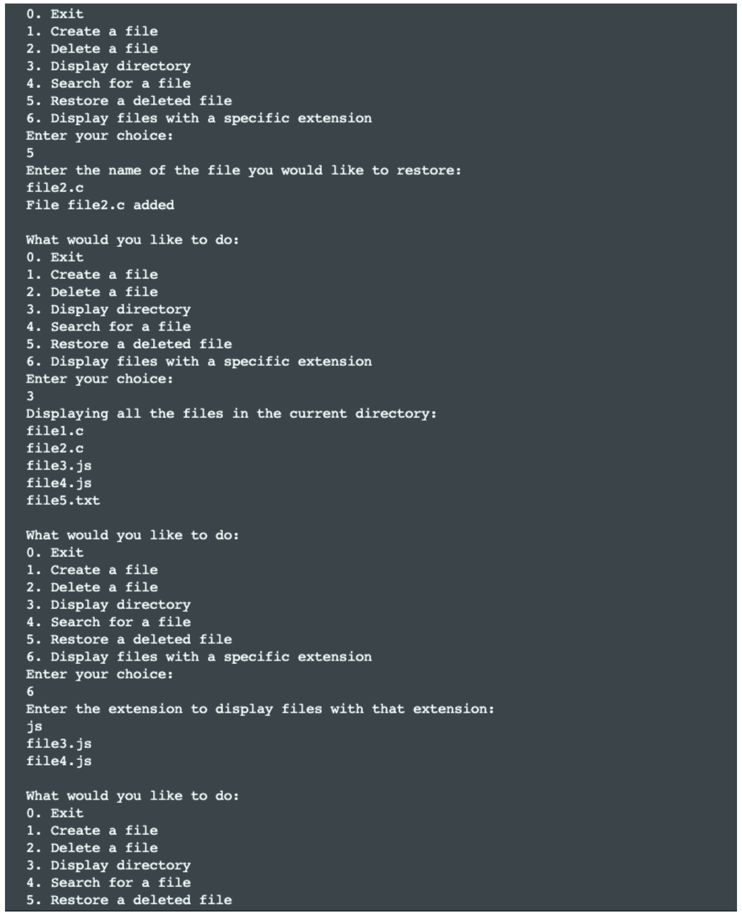


```
Enter the name of the file that you would like to create:
file4.js
File file4.js added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
1
Enter the name of the file that you would like to create:
file5.txt
File file5.txt added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
2
Enter the name of the file that you would like to delete:
file4.js
File file4.js deleted

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
2
Enter the name of the file that you would like to delete:
file2.c
File file2.c deleted

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
```

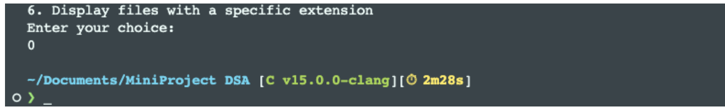
Output Screenshots



```
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
3
Displaying all the files in the current directory:
file1.c
file3.js
file5.txt

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
4
Enter the name of the file you wish to search for:
file3.js
File found: file3.js

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
4
Enter the name of the file you wish to search for:
file4.js
File not found: file4.js

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
5
Enter the name of the file you would like to restore:
file4.js
File file4.js added

What would you like to do:
```

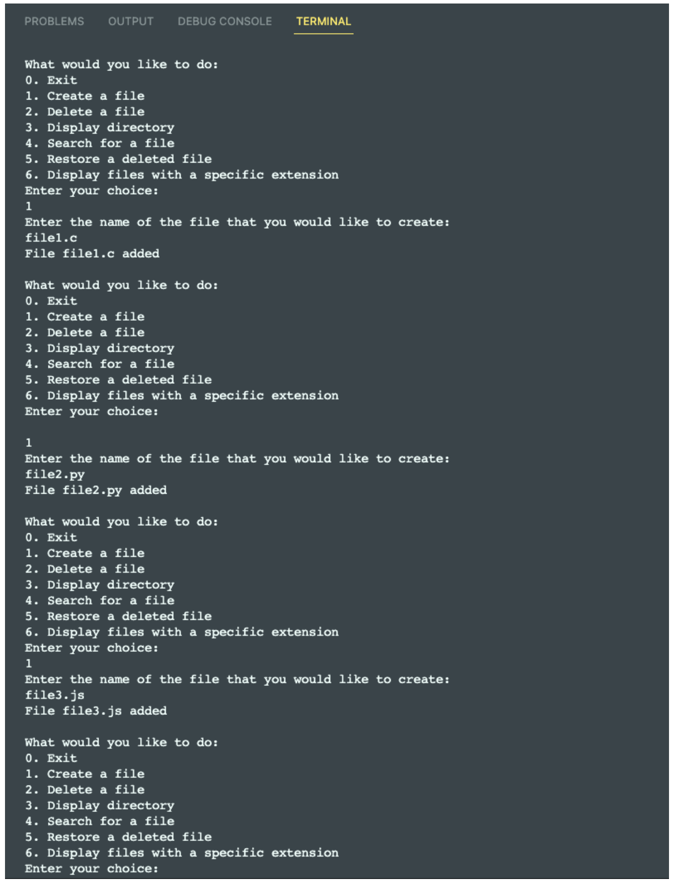


```
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
5
Enter the name of the file you would like to restore:
file2.c
File file2.c added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
3
Displaying all the files in the current directory:
file1.c
file2.c
file3.js
file4.js
file5.txt

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
6
Enter the extension to display files with that extension:
js
file3.js
file4.js

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
```

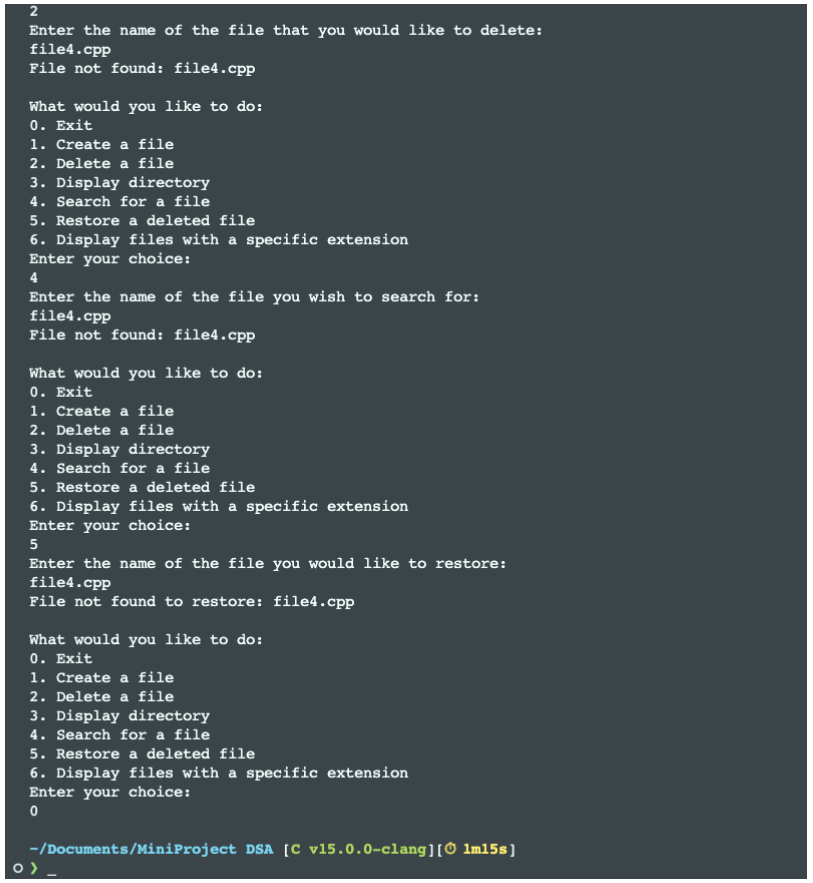


```
6. Display files with a specific extension
Enter your choice:
0

~/Documents/MiniProject DSA [C v15.0.0-clang][⏱ 2m28s]
○ ❯ _
```

Output Screenshots Showing Error Handling



```
What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
1
Enter the name of the file that you would like to create:
file1.c
File file1.c added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:

1
Enter the name of the file that you would like to create:
file2.py
File file2.py added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
1
Enter the name of the file that you would like to create:
file3.js
File file3.js added

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
```



```
2
Enter the name of the file that you would like to delete:
file4.cpp
File not found: file4.cpp

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
4
Enter the name of the file you wish to search for:
file4.cpp
File not found: file4.cpp

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
5
Enter the name of the file you would like to restore:
file4.cpp
File not found to restore: file4.cpp

What would you like to do:
0. Exit
1. Create a file
2. Delete a file
3. Display directory
4. Search for a file
5. Restore a deleted file
6. Display files with a specific extension
Enter your choice:
0

~/Documents/MiniProject DSA [C v15.0.0-clang][⏱ 1m15s]
○ ❯ _
```

Contribution of each Team Member

1. Nannapaneni Sashank
- Creating File
- Deleting File

2. Nikhil Upadhyay
- Searching File
- Displaying all files in the directory

3. Nishank Bhowal
- Viewing files with particular extension

4. Nishanth J Modpur
- Restoring deleted files