PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester :** I

Name of the student: Samruddhi Ramswami Bansode. **PRN :** 124B1B019

Department: Computer Engineering **Division :** A

Course Name : Data Structures

Course Code: BCE23PC02

Completion Date : 25/08/2025

Assignment No. 6

Problem Statement:

File navigation system

create a file navigation system that stimulate forward and backward navigation similar to file explorers or web browsers

- Key operation to be perform :
  navigate to a new folder (add node)
  go back (move left in DLL)
  go forward (move right)
  display current path

Source Code :

```
#include <iostream>

using namespace std;



class Node {

public:

    string folder;

    Node* prev;
```

```
    Node* next;


    Node(string f) {
        folder = f;
        prev = next = NULL;
    }
};


class FileNavigator {
private:
    Node* current;


public:
    FileNavigator(string fol) {
        current = new Node(fol);
    }


    // navigate to new folder
    void navigate(string folderName) {
        Node* newNode = new Node(folderName);


        // clear forward history
        if (current->next != NULL) {
            Node* temp = current->next;
            while (temp != NULL) {
                Node* toDelete = temp;
                temp = temp->next;
```

```
            delete toDelete;
        }
        current->next = NULL;
    }

    newNode->prev = current;
    current->next = newNode;
    current = newNode;
}

// go backward
void goBack() {
    if (current->prev != NULL) {
        current = current->prev;
    } else {
        cout << "No previous folder!\n";
    }
}

// go forward
void goForward() {
    if (current->next != NULL) {
        current = current->next;
    } else {
        cout << "No forward folder!\n";
    }
}
```

```
    // show current path
    void showPath() {
        Node* temp = current;
        while (temp->prev != NULL) {
            temp = temp->prev;
        }

        cout << "Current Path: ";
        while (temp != NULL) {
            cout << "/" << temp->folder;
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    FileNavigator nav("C:");
    int choice;
    string folder;
    cout << "\n--- File Navigation Menu ---\n";
    cout << "1. Navigate to new folder\n";
    cout << "2. Go Back\n";
    cout << "3. Go Forward\n";
    cout << "4. Show Current Path\n";
    cout << "5. Exit\n";
```

```
do {
    cout << "Enter choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            cout << "Enter folder name: ";
            cin >> folder;
            nav.navigate(folder);
            break;
        case 2:
            nav.goBack();
            break;
        case 3:
            nav.goForward();
            break;
        case 4:
            nav.showPath();
            break;
        case 5:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice!\n";
    }
} while (choice != 5);
```

```
    return 0;
}
```

Screen Shot of Output :

```
PS C:\C++ DSA> g++ assig6.cpp
PS C:\C++ DSA> ./a.exe

--- File Navigation Menu ---
1. Navigate to new folder
2. Go Back
3. Go Forward
4. Show Current Path
5. Exit
Enter choice: 1
Enter folder name: dsa
Enter choice: 1
Enter folder name: unit1
Enter choice: 1
Enter folder name: quicksort
Enter choice: 4
Current Path: /C:/dsa/unit1/quicksort
Enter choice: 1
Enter folder name: projects
Enter choice: 2
Enter choice: 4
Current Path: /C:/dsa/unit1/quicksort/projects
Enter choice: 3
Enter choice: 4
Current Path: /C:/dsa/unit1/quicksort/projects
Enter choice: 5
Exiting...
PS C:\C++ DSA>
```

Conclusion:

The problem is solved using the concept of a **doubly linked list**, where each node stores a folder name and links to previous and next folders. Operations like navigating to a new folder, moving backward, moving forward, and displaying the current path are implemented by updating the current pointer in the list. This approach provides an efficient way to simulate file navigation similar to browsers or file explorers.