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: Om Jitendra Khalane **PRN :** 124B1B040

Department: Computer Engineering **Division :** A

Course Name : Data Structures and Laboratory

Course Code: BCE23PC02

Completion Date : 12/08/2025

Problem Statement:

"Browser History Management"

You are tasked with implementing a simplified browser history management system. This system should allow users to navigate between previously visited pages, and also add new pages to their history. The system needs to efficiently handle both "back" and "forward" navigation, as well as adding new pages, while maintaining a limited history size.

Requirements:

When a user visits a new page, it should be added to the current history. If there are any "forward" pages available (i.e., pages visited after the current page in the history), they should be cleared, as visiting a new page effectively creates a new branch in the history.

Users should be able to navigate to the previous page in their history. If there's no previous page, they remain on the current page.

Users should be able to navigate to a page they previously went "back" from. If there's no page to go forward to, they remain on the current page.

The browser history should have a maximum capacity. If adding a new page exceeds this capacity, the oldest page in the history should be automatically removed to make space.

Input:

A sequence of operations:

visit(url): Adds url to the history.

back(steps): Navigates back by steps pages.

forward(steps): Navigates forward by steps pages.

Output:

For each visit, back, or forward operation, output the URL of the page the user is currently on after the operation.

Source Code :

https://github.com/omkhalane/DSAL-SY-PCCOE/blob/main/lab_assignments/october_challenge.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

class BrowserHistory
{
private:
    int capacity;
    string current;
    deque<string> backStack;
    stack<string> forwardStack;

public:
    BrowserHistory(int cap, string homepage)
    {
        capacity = cap;
        current = homepage;
    }

    void visit(const string &url)
```

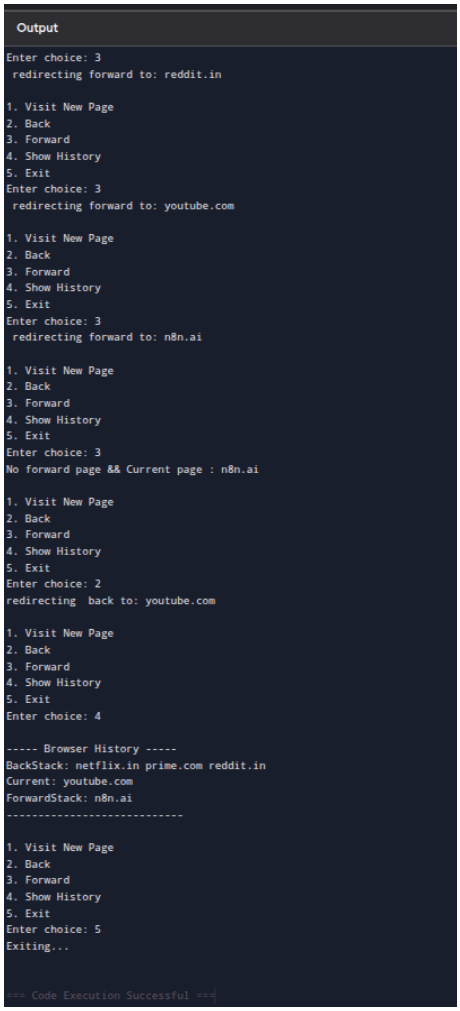
```
        {
            backStack.push_back(current);

            if ((int)backStack.size() > capacity)
            {
                backStack.pop_front();
            }

            while (!forwardStack.empty())
            {
                forwardStack.pop();
            }

            current = url;
            cout << "Visited: " << current << endl;
        }

        void back()
        {
            if (backStack.empty())
            {
                cout << "No previous page && Current Page : " << current <<
endl;
                return;
            }

            forwardStack.push(current);
            current = backStack.back();
            backStack.pop_back();

            cout << "redirecting  back to: " << current << endl;
        }
```

```
    void forward()
    {
        if (forwardStack.empty())
        {
            cout << "No forward page && Current page : " << current << endl;
            return;
        }

        backStack.push_back(current);
        current = forwardStack.top();
        forwardStack.pop();

        if ((int)backStack.size() > capacity)
        {
            backStack.pop_front();
        }

        cout << " redirecting forward to: " << current << endl;
    }

    // Display
    void showHistory()
    {
        cout << "\n----- Browser History -----\n";
        cout << "BackStack: ";
        for (auto &p : backStack)
            cout << p << " ";
        cout << "\nCurrent: " << current;
        cout << "\nForwardStack: ";

        stack<string> temp = forwardStack;
```

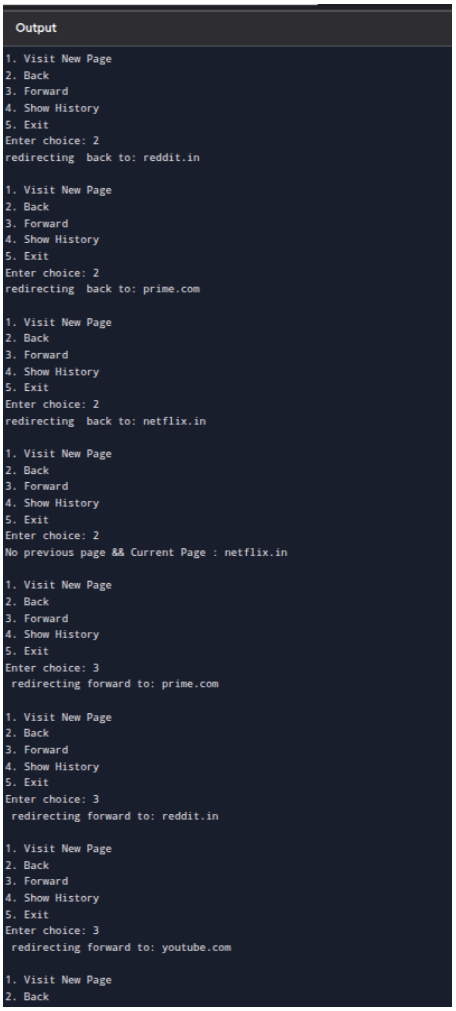
```
            while (!temp.empty())
            {
                cout << temp.top() << " ";
                temp.pop();
            }
            cout << "\n----------------------------\n";
        }
    };

    // Driver code
    int main()
    {
        int capacity;
        cout << "Enter browser history capacity: ";
        cin >> capacity;

        string homepage;
        cout << "Enter homepage URL: ";
        cin >> homepage;

        BrowserHistory browser(capacity, homepage);

        int choice;
        string url;

        do
        {
            cout << "\n1. Visit New Page\n2. Back\n3. Forward\n4. Show
History\n5. Exit\nEnter choice: ";
            cin >> choice;

            switch (choice)
```

```
            {
            case 1:
                cout << "Enter URL: ";
                cin >> url;
                browser.visit(url);
                break;

            case 2:
                browser.back();
                break;

            case 3:
                browser.forward();
                break;

            case 4:
                browser.showHistory();
                break;

            case 5:
                cout << "Exiting...\n";
                break;

            default:
                cout << "Invalid choice! Try again.\n";
            }

        } while (choice != 5);

        return 0;
    }
```

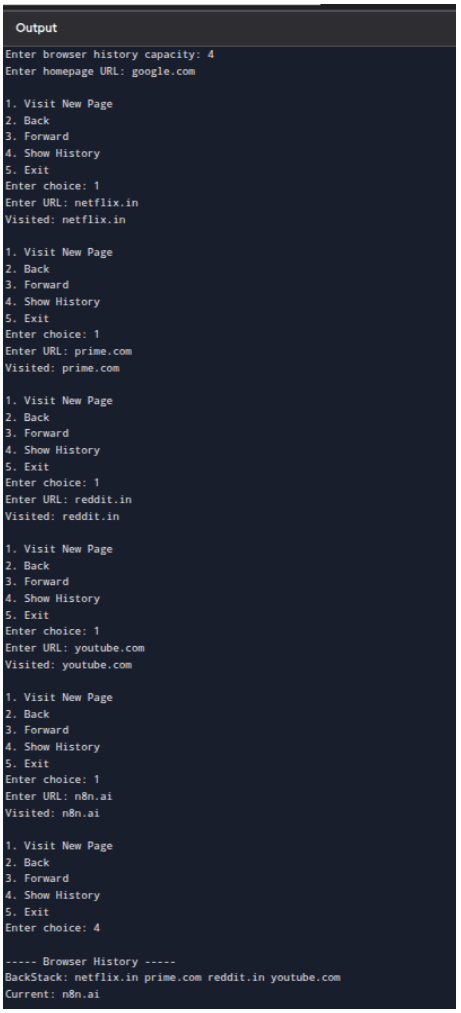
Output :



```
Output

Enter choice: 3
 redirecting forward to: reddit.in

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 3
 redirecting forward to: youtube.com

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 3
 redirecting forward to: n8n.ai

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 3
No forward page && Current page : n8n.ai

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 2
redirecting  back to: youtube.com

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 4

----- Browser History -----
BackStack: netflix.in prime.com reddit.in
Current: youtube.com
ForwardStack: n8n.ai
---------------------------

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 5
Exiting...


=== Code Execution Successful ===
```



```
Output

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 2
redirecting  back to: reddit.in

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 2
redirecting  back to: prime.com

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 2
redirecting  back to: netflix.in

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 2
No previous page && Current Page : netflix.in

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 3
 redirecting forward to: prime.com

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 3
 redirecting forward to: reddit.in

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 3
 redirecting forward to: youtube.com

1. Visit New Page
2. Back
```



```
Output

Enter browser history capacity: 4
Enter homepage URL: google.com

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 1
Enter URL: netflix.in
Visited: netflix.in

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 1
Enter URL: prime.com
Visited: prime.com

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 1
Enter URL: reddit.in
Visited: reddit.in

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 1
Enter URL: youtube.com
Visited: youtube.com

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 1
Enter URL: n8n.ai
Visited: n8n.ai

1. Visit New Page
2. Back
3. Forward
4. Show History
5. Exit
Enter choice: 4

----- Browser History -----
BackStack: netflix.in prime.com reddit.in youtube.com
Current: n8n.ai
```

Conclusion:

The **Browser History Management System** simulates how modern browsers maintain navigation history using data structures like **stack** and **deque**.

- The **back stack** stores URLs of pages visited before the current page.
- The **forward stack** stores pages that can be revisited when the user navigates forward.
- The **current page** represents the active webpage the user is viewing.

- The forward stack is cleared to remove any “future” history.
- If the back stack exceeds the defined **capacity**, the oldest page is automatically removed to maintain memory efficiency.
- **Deque (Double-ended Queue):** Used for the back stack to manage both ends efficiently (adding/removing from front and back).
- **Stack:** Used for forward navigation since it allows Last-In-First-Out (LIFO) access.

Time Complexity (TC):

- **Visit:** O(1)
- **Back:** O(1)
- **Forward:** O(1)
- **Show History:** O(n) (to traverse and display the stacks)

Space Complexity (SC):

- **O(n)** where n = number of URLs stored within capacity limit.