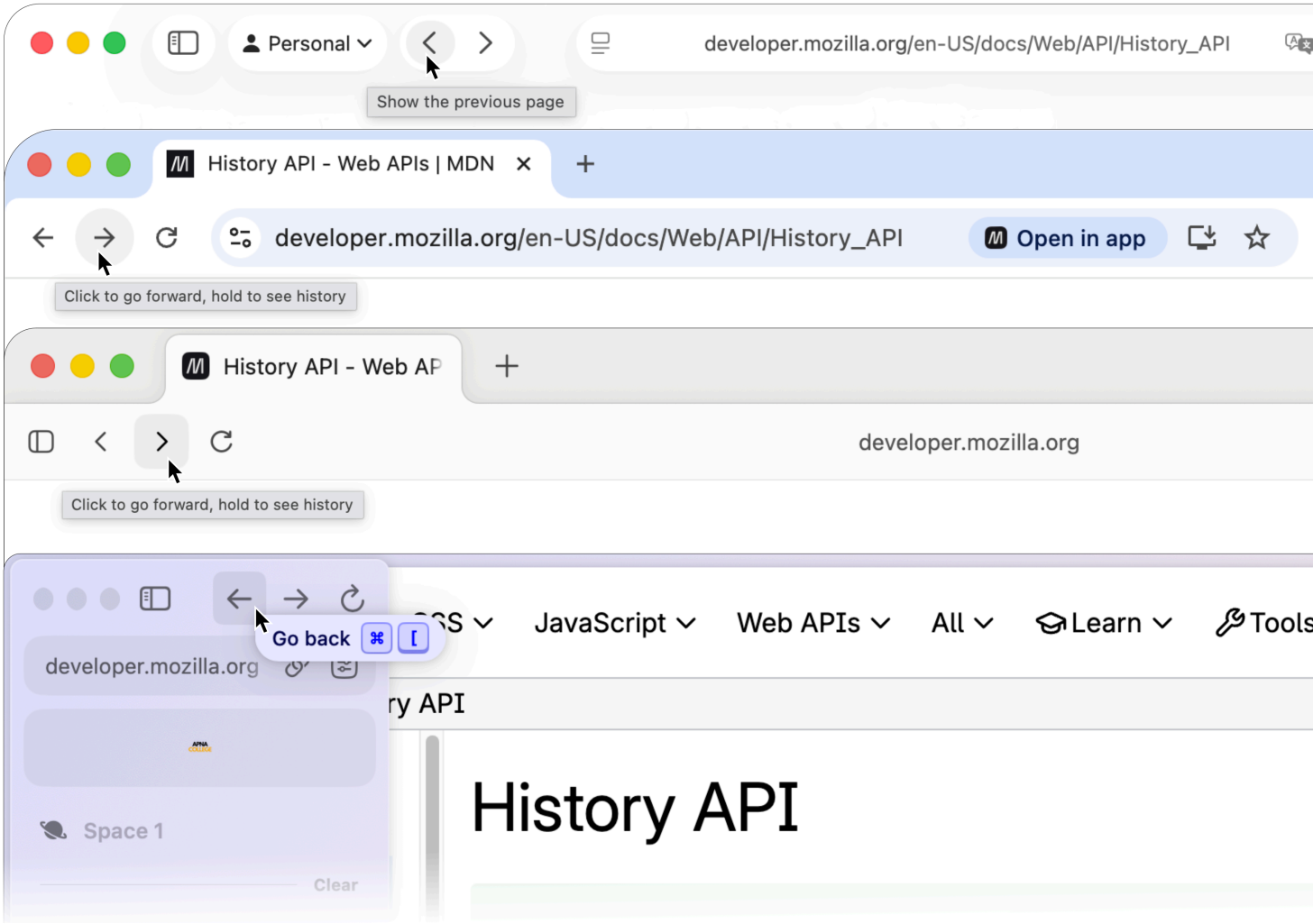
Project Report

Browser History Navigation



Lorem ipsum dolor
Sed vitae ligula vel justo gravida dignissim
Curabitur euismod mauris ut ipsum sodales, nec commodo erat tincid

Prepared by,
primit
2 April 2002

Table of Contents

Problem Statement Analysis

This project implements a simplified Browser History Navigation System in the C programming language. It simulates the essential navigation features of real web browsers - `back()`, `forward()`, `go()`, and `history.length()` — using two stacks to store backward and forward navigation paths. The goal is to demonstrate stack-based navigation logic in a console-based environment.

The project belongs to the domain of **browser navigation systems** and **data structure applications**. It focuses specifically on how browsers internally manage navigation through stack operations. The model excludes rendering, networking, and caching, and instead concentrates on the algorithmic mechanism that enables backward and forward movement between visited URLs.

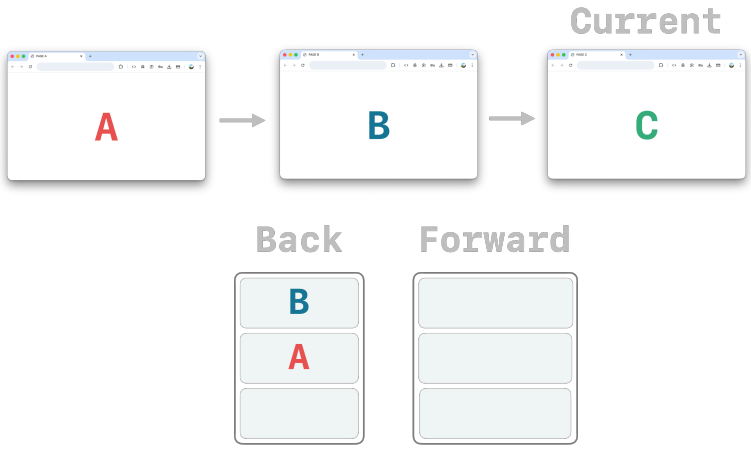
This problem requires analyzing:

- How to store past and future navigation states
- How to efficiently update history during Back/Forward events
- How to ensure constant-time operations for all navigation actions

Real World Browser Analogy

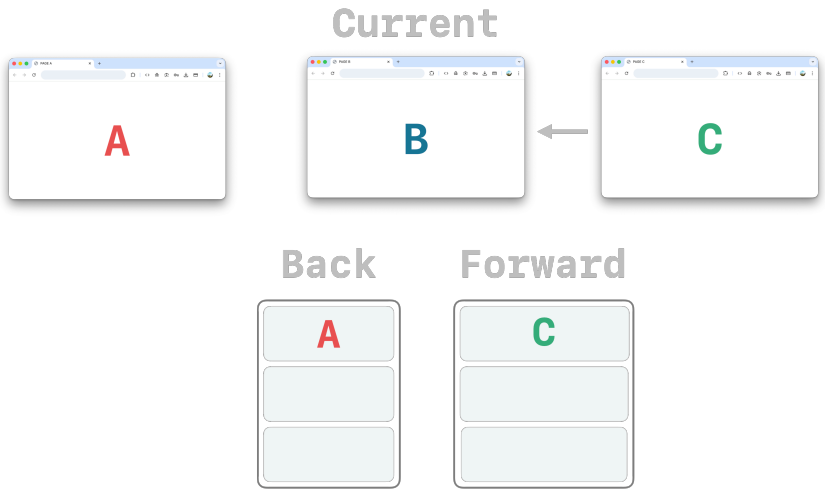
Real browsers internally maintain two stacks: a Back Stack and a Forward Stack. These stacks change dynamically based on how the user navigates between pages.

Visiting New Pages



- Forward Stack stays empty
- Each new visit pushes the previous page to the Back Stack and clears the Forward Stack.

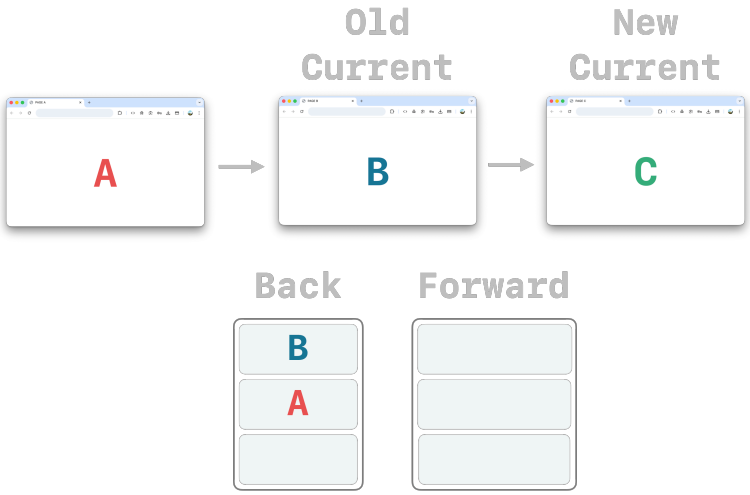
Going Back



If the user goes Back on page C → B:

- C is pushed to the Forward Stack
- B is popped from the Back Stack and becomes the current page
- Back Stack: [A]
- Forward Stack: [C]

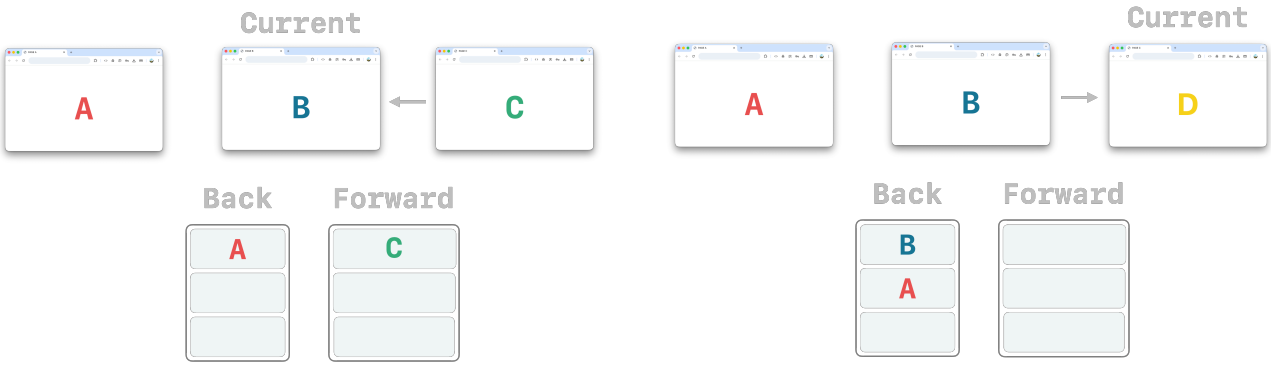
Going Forward



If the user presses Forward B → C:

- B is moved to the Back Stack
- C is popped from the Forward Stack

- Forward Stack becomes empty again



Going Back and Then Visiting a New Page

- If the user goes C → (Back) → B and then visits D:
- B goes to Back Stack
- Forward Stack is cleared because a new visit breaks the previous forward path
- Final stacks:
- Back Stack: [A, B]
- Forward Stack: empty
- Current page: D

Applicable Data Structure Exploration

Browser history requires three core behaviors:

- Record the current page when navigating to a new one
- Undo the most recent navigation (Back)
- Redo an undone navigation (Forward)

To support these actions efficiently, the data structure must allow quick access to the most recently visited items.

Arrays and linked lists can store history, but they don't map cleanly to the Back and Forward concepts:

Navigating back means moving a pointer backward; navigating forward means moving it forward.

- This works, but clearing the forward path on a new visit requires multiple operations.
- Arrays require shifting elements when trimming forward history.
- Linked lists avoid shifting, but managing previous/next pointers adds complexity with no real benefit.

Overall, these structures behave like a linear timeline, but they do not model the undo/redo pattern efficiently.

Why Stacks are a better fit?

Stacks match the navigation model directly:

- Back is an undo → pop from Back Stack, push to Forward Stack
- Forward is a redo → pop from Forward Stack, push to Back Stack
- New Visit invalidates future states → clear Forward Stack

All operations—push, pop, and clear—are O(1), and the Last-In-First-Out behavior corresponds exactly to how browsers manage recent navigation.

This is why most simplified browser-history models, including this project, use a two-stack approach: one stack for past pages (Back), and one stack for future pages (Forward).

Designed / Proposed Solution / Algorithm

This section describes how the proposed two-stack browser navigation model is implemented in C. It covers the data structures, function-level behavior, key logic, and the flow of control within the program. The implementation stays close to the design discussed earlier, ensuring that each navigation action behaves like a real browser.

Data Structure Implementation

Stack Structure

Each stack is implemented as a simple array-based structure:

- items[][] stores URLs
- top tracks the index of the most recent entry

```
typedef struct {
    char items[STACK_SIZE][MAX_URL_LEN];
    int top;
} Stack;
```

Array-based stacks are chosen because:

- They allow O(1) push/pop operations
- No dynamic memory allocation is required
- Index management is simple and predictable

BrowserHistory Structure

The BrowserHistory structure integrates all components:

```
typedef struct {
    Stack backStack;
    Stack forwardStack;
    char currentPage[MAX_URL_LEN];
} BrowserHistory;
```

It contains:

- backStack → pages behind the current page
- forwardStack → pages ahead of the current page
- currentPage → the page being viewed right now

This layout is directly aligned with real browser navigation internals.

Initialization Logic

The browser starts with:

- A homepage
- Empty back and forward stacks

```
void initBrowser(BrowserHistory *b, const char
*homepage) {
    b→backStack.top = -1;
    b→forwardStack.top = -1;
    strcpy(b→currentPage, homepage);
}
```

By setting top = -1, both stacks start empty.

Core Function Implementations

`visit()`

Implements the algorithm for loading a new page.

```
void visit(BrowserHistory *b, const char *url) {
    push(&b→backStack, b→currentPage); // Save
current page to back history
    strcpy(b→currentPage, url); // New page becomes
current
    clear(&b→forwardStack); // Forward history is
```

Behavior:

- Saves the previous page
- Replaces the current page
- Clears the forward stack (matching Chrome/Firefox behavior)

`back()`

Moves backward through navigation history.

```
void back(BrowserHistory *b, int steps) {
    for(int i = 0; i < steps; i++) {
        push(&b→forwardStack, b→currentPage);
        strcpy(b→currentPage, pop(&b→backStack));
    }
}
```

Behavior

- Current page → forward stack
- Back stack top → current page
- Repeats for steps times
- Stops automatically if history is insufficient

This is equivalent to repeatedly pressing the Back button in a browser.

`forward()`

Moves forward through the history.

```
void forward(BrowserHistory *b, int steps) {
    for(int i = 0; i < steps; i++) {
        push(&b→backStack, b→currentPage);
        strcpy(b→currentPage, pop(&b
>forwardStack));
    }
`
```

Behavior

- Current page → back stack
- Forward stack top → current page
- Repeats for steps

This restores previously undone pages.

`showHistory()`

The system includes a showHistory() function that prints the entire navigation state in a readable format. Instead of returning just the length of the history, this function displays:

- All pages stored in the Back Stack
- The Current Page
- All pages stored in the Forward Stack

This provides a clear picture of how navigation changes over time.

How it works:

1. The function iterates from the bottom to the top of the Back Stack and prints past pages.
2. It prints the current page with index 0.
3. It then iterates through the Forward Stack (top to bottom) to show possible forward navigation.

This layout visually represents:

```
(backwards)  -3  -2  -1   0   +1   +2  (forwards)
```

Example Output:

```
-2. homepage.com
-1. page1.com
 0. page2.com   ⟵ current page
 1. page3.com
```

Purpose:

This function acts as a human-readable version of history.length(), since it shows:

- Total history size
- Page order
- Current page position
- Forward/backward navigation depth

It helps users understand the behavior of the two-stack model during testing.

Stack Operations

These helper functions support all navigation logic.

`push()`

Adds a new page onto the specified stack:

```
void push(Stack *s, const char *url) {
    s→top++;
    strcpy(s→items[s→top], url);
}
```

`pop()`

Returns the most recent page:

```
char* pop(Stack *s) {
    return s->items[s->top--];
}
```

clear()

Used in visit() to clear forward navigation:

```
void clear(Stack *s) {
    s->top = -1;
}
```

The reset is O(1) and does not require deleting elements.

User Interface Logic

The UI is a simple menu-driven console program using ASCII art.
It displays the current page and offers:

1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit

Render:

```
┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: homepage.com           │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
│ 2. Go back                           │
│ 3. Go forward                        │
│ 4. Show browser history              │
│ 5. Exit                              │
└──────────────────────────────────────┘
```

This allows the user to interactively test stack-based navigation.

Flow of Execution

A simplified flow:

```
User selects action
      ↓
Input collected
      ↓
Corresponding function called
      ↓
Stacks updated based on action
      ↓
Current page updated
      ↓
Result printed to console
```

This loop continues until the user chooses to exit.

This matches the behavior of Chrome/Edge/Firefox.

Implementation of Proposed Solution / Algorithm

The browser history system operates in a continuous loop, where each cycle processes one complete user action. The execution flow ensures that user input is handled safely, the correct navigation function is called, and the internal stacks are updated accordingly. The detailed flow of execution is as follows:

1. User selects an action

The program first displays a menu containing all available operations (Visit, Back, Forward, Show History, Exit).
The user enters a choice, which the program reads using scanf().

2. Input is collected

Based on the user's selection, the program gathers the required additional input:

- For Visit, the program reads the URL using fgets().
- For Back or Forward, the program reads the number of steps.

Input validation is performed to ensure that invalid entries do not break the loop.

3. Corresponding function is called

Once valid input is received, the program invokes the appropriate function:

- visit(&browser, url)
- back(&browser, steps)
- forward(&browser, steps)
- showHistory(&browser)

Each function directly follows the two-stack navigation algorithm designed earlier.

4. Stacks are updated based on the action

Inside the called function:

- Pages are pushed or popped from Back Stack and Forward Stack.
- Visiting a new page clears the Forward Stack.
- Back and Forward actions transfer the current page between the stacks.
- All updates occur using constant-time stack operations (push, pop, clear).

5. Current page is updated

After each operation:

- The value of browser.currentPage is replaced with the appropriate URL
- (either from the user input or popped from a stack).
- This variable always reflects the page the user is "currently on."

6. Result is printed to the console

Each function prints feedback:

- "Visited: ..."
- "Went back to: ..."
- "Went forward to: ..."
- Or a formatted history view via showHistory().

This ensures the user always sees the updated state of the browser history.

7. Loop continues

After displaying the result:

- Control returns to the menu and the user is prompted again.
- The loop only terminates when the user selects the Exit option.

Main Program and Logic (BrowserHistory.c)

```
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>.

#define MAX_URL_LEN 256
#define STACK_SIZE 100

// Stack Structure
typedef struct {
    char items[STACK_SIZE][MAX_URL_LEN];
    int top;
} Stack;

// Stack Functions
void push(Stack *s, const char *url) {
    if (s->top < STACK_SIZE -1) {
        s->top++;
        strcpy(s->items[s->top], url);
    }
}

char* pop(Stack *s) {
    if(s->top >= 0) {
        return s->items[s->top--];
    }
    return NULL;
}

char* peek(Stack *s) {
    if(s->top >= 0) {
        return s->items[s->top];
    }
    return NULL;
}

int isEmpty(Stack *s) {
    return s->top == -1;
}

void clear(Stack *s) {
    s->top = -1;
}

// Browser History struct
```

```
typedef struct {
    Stack backStack;
    Stack forwardStack;
    char currentPage[MAX_URL_LEN];
} BrowserHistory;

// Initalize
void initBrowser(BrowserHistory *b, const char *homepage) {
    b→backStack.top = -1;
    b→forwardStack.top = -1;
    strcpy(b→currentPage, homepage);
}

// Visit function
void visit(BrowserHistory *b, const char *url) {
    // Only push to history if visiting a different page
    if (strcmp(b→currentPage, url) ≠ 0) {
        push(&b→backStack, b→currentPage);
        strcpy(b→currentPage, url);
        clear(&b→forwardStack);
        printf("\nVisited: %s", url);
    } else {
        printf("\nAlready on: %s (page reloaded)", url);
    }
}

// Back function
void back(BrowserHistory *b, int steps) {
    int available = b→backStack.top + 1; // how many pages
in back stack

    if( steps ⩽ 0 || steps > available) {
        printf("\nCannot go back %d steps.", steps);
        return;
    }

    char *url;
    for(int i=0; i< steps; i++) {
        if(!isEmpty(&b→backStack)) {
            push(&b→forwardStack, b→currentPage);
            url = pop(&b→backStack);
            strcpy(b→currentPage, url);
        }
```

```
    }
    printf(“\nWent back to: %s\n", b→currentPage);
}

// Forward function
void forward(BrowserHistory *b, int Steps) {

    int available = b→forwardStack.top + 1; // how many
pages in forward stack

    if (Steps ⩽ 0 || Steps > available) {
        printf(“\nCannot go forward %d steps.", Steps);
        return;
    }

    char *url;
    for(int i=0; i < Steps; i++) {
        if(!isEmpty(&b→forwardStack)) {
            push(&b→backStack, b→currentPage);
            url = pop(&b→forwardStack);
            strcpy(b→currentPage, url);
        }
    }
    printf(“\nWent forward to: %s\n", b→currentPage);
}

// Show History Function
void showHistory(BrowserHistory *b) {
    printf("\n======== Browser Stack View ========\n");

    Stack *back = &b→backStack;
    Stack *forward = &b→forwardStack;

    // backward stack
    for(int i = 0; i ⩽ back→top; i++) {
        int index = -(back→top - i + 1);
        printf("%d. %s\n", index, back→items[i]);
    }

    // current page
    printf("0. %s  ⟵ current page\n", b→currentPage);

    // forward stack
```

```
    for(int i = forward→top; i ⩾ 0; i--) {
        int index = (forward→top - i) + 1;
        printf("%d. %s\n", index, forward→items[i]);
    }
    printf("====================================\n");
}

// Main
int main(void) {
    BrowserHistory browser;
    initBrowser(&browser, "google.com");

    int choice;
    char url[MAX_URL_LEN];

    while (1) {

        printf(“\n);


printf("\n╔═══════════════════════════════════════════════╗\n");
        printf("║                  BROWSER
║\n");

printf("╠═══════════════════════════════════════════════╣\n");
        printf("║ Current Page: %-25s ║\n",
browser.currentPage);

printf("╠═══════════════════════════════════════════════╣\n");
        printf("║ 1. Visit new page
║\n");
        printf("║ 2. Go back
║\n");
        printf("║ 3. Go forward
║\n");
        printf("║ 4. Show browser history
║\n");
        printf("║ 5. Exit
║\n");

printf("╚═══════════════════════════════════════════════╝\n");
        printf("Choose Option: ");
```

```
        if (scanf("%d", &choice) != 1) {
            // Clear invalid input from buffer
            while (getchar() != '\n');
            printf("\nInvalid option. Please enter a number.
\n");
            continue;
        }

        getchar(); // clear newline

        switch (choice) {
            case 1: {
                printf("Enter url: ");
                fgets(url, MAX_URL_LEN, stdin);
                url[strcspn(url, "\n")] = 0; // remove
newline
                visit(&browser, url);
                break;
            }
            case 2: {
                int steps;
                printf("Enter Steps: ");
                scanf("%d", &steps);
                back(&browser, steps);
                break;
            }
            case 3: {
                int steps;
                printf("Enter Steps: ");
                scanf("%d", &steps);
                forward(&browser, steps);
                break;
            }
            case 4: {
                showHistory(&browser);
                break;
            }

            case 5: {
                printf("\nExiting Browser....\n");
                return 0;
            }
```

```
            default: {
                printf("\nInvalid option.\n");
            }
        }
    }

    return 0;
}
```

Testing

Test 01 - Back navigation (valid steps)

Description: Tests whether the system correctly goes back a valid number of steps
Items Covered: back(), push(), pop(), stack transitions.
Requirements Addressed: Browser must support the multi-step backward navigation.
Environment: Stand C computer (GCC).
Dependencies: Must have visited at least 2 pages.

1. Visit `page1.com`, then `page2.com`, then `page3.com`
2. Call back 2 Steps
3. Check the resulting page

Input Specification: Steps = 2
Expected Output: Current page should become Page A, forward stack should contain B and C.
Pass/Fail Criteria: System returns the correct page and updates both stacks accordingly.

```
BROWSER
Current Page: google.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
```

```
5. Exit
```

```
Choose Option: 1
Enter url: page1.com

Visited: page1.com
```

```
                BROWSER
Current Page: page1.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: page2.com

Invalid option. Please enter a number.
```

```
                BROWSER
Current Page: page1.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page2.com

Visited: page2.com
```

```
                BROWSER
Current Page: page2.com
```

```
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page3.com

Visited: page3.com
```

```
                BROWSER
Current Page: page3.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 2

Went back to: page1.com
```

```
                BROWSER
Current Page: page1.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 4

======== Browser Stack View ========
-1. google.com
0. page1.com  ⟵ current page
```

```
1. page2.com
2. page3.com
==================================
```

Test 02 - Back Navigation (Invalid Steps)

Description: Ensure invalid backward steps are rejected.
Items Covered: back() validation.
Requirements Addressed: Error handling for invalid inputs.
Environment: Standard terminal environment.
Dependencies: None.
Test Procedure:

1. Visit one page.
2. Call back(browser, 3) (more than available).

Input Specification: Steps = 3
Expected Output: Message: "Cannot go back 3 steps."
Pass/Fail Criteria: System does not crash and returns an error message.

```
╔═══════════════════════════════════╗
║              BROWSER              ║
╠═══════════════════════════════════╣
║ Current Page: google.com          ║
╠═══════════════════════════════════╣
║ 1. Visit new page                 ║
║ 2. Go back                        ║
║ 3. Go forward                     ║
║ 4. Show browser history           ║
║ 5. Exit                           ║
╚═══════════════════════════════════╝
Choose Option: 1
Enter url: page1.com

Visited: page1.com

╔═══════════════════════════════════╗
║              BROWSER              ║
╠═══════════════════════════════════╣
║ Current Page: page1.com           ║
╠═══════════════════════════════════╣
║ 1. Visit new page                 ║
```

```
| 2. Go back
| 3. Go forward
| 4. Show browser history
| 5. Exit
```

```
Choose Option: 2
Enter Steps: 3

Cannot go back 3 steps.
```

```
+-----------------------------------+
|              BROWSER              |
+-----------------------------------+
| Current Page: page1.com           |
+-----------------------------------+
| 1. Visit new page                 |
| 2. Go back                        |
| 3. Go forward                     |
| 4. Show browser history           |
| 5. Exit                           |
+-----------------------------------+
```

```
Choose Option: 4

======== Browser Stack View ========
-1. google.com
0. page1.com  ⟵ current page
====================================
```

Test 03 - Forward Navigation (Valid Steps)

Description: Verifies correct forward movement after going backward.
Items Covered: forward(), stack transitions.
Requirements Addressed: Multi-step forward navigation.

Test Procedure:
1. Visit three pages.
2. Go back 2 steps.
3. Call forward 1 steps.

Input Specification: Steps = 1

Expected Output: The New current page should be the last page popped from the forward stack.
Pass/Fail Criteria: Forward stack decreases, back stack increases, and the correct page is displayed.

```
┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: google.com             │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
│ 2. Go back                           │
│ 3. Go forward                        │
│ 4. Show browser history              │
│ 5. Exit                              │
└──────────────────────────────────────┘
Choose Option: 1
Enter url: page1.com

Visited: page1.com

┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: page1.com              │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
│ 2. Go back                           │
│ 3. Go forward                        │
│ 4. Show browser history              │
│ 5. Exit                              │
└──────────────────────────────────────┘
Choose Option: 2
Enter Steps: 3

Cannot go back 3 steps.

┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: page1.com              │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
```

```
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 4

======== Browser Stack View ========
-1. google.com
0. page1.com  ⟵ current page
====================================
```

```
BROWSER
Current Page: page1.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 1

Went back to: google.com
```

```
BROWSER
Current Page: google.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page1.com
```

```
Visited: page1.com
```

```
╔═══════════════════════════════════════╗
║                BROWSER                ║
╠═══════════════════════════════════════╣
║ Current Page: page1.com               ║
╠═══════════════════════════════════════╣
║ 1. Visit new page                     ║
║ 2. Go back                            ║
║ 3. Go forward                         ║
║ 4. Show browser history               ║
║ 5. Exit                               ║
╚═══════════════════════════════════════╝
Choose Option: 1
Enter url: page2.com

Visited: page2.com
```

```
╔═══════════════════════════════════════╗
║                BROWSER                ║
╠═══════════════════════════════════════╣
║ Current Page: page2.com               ║
╠═══════════════════════════════════════╣
║ 1. Visit new page                     ║
║ 2. Go back                            ║
║ 3. Go forward                         ║
║ 4. Show browser history               ║
║ 5. Exit                               ║
╚═══════════════════════════════════════╝
Choose Option: 1
Enter url: page3.com

Visited: page3.com
```

```
╔═══════════════════════════════════════╗
║                BROWSER                ║
╠═══════════════════════════════════════╣
║ Current Page: page3.com               ║
╠═══════════════════════════════════════╣
║ 1. Visit new page                     ║
║ 2. Go back                            ║
║ 3. Go forward
```

```
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 2

Went back to: page1.com
```

```
BROWSER
Current Page: page1.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 3
Enter Steps: 1

Went forward to: page2.com
```

```
BROWSER
Current Page: page2.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 4

======== Browser Stack View ========
-2. google.com
-1. page1.com
0. page2.com  ⟵ current page
```

```
1. page3.com
==================================
```

Test 04 - Visiting a new page resets the forward stack

Description: Tests clearing forward history after visiting a new page.
Items Covered: visit(), clear().

Test Procedure:
1. Visit `page1.com`, then `page2.com`, then `page3.com`
2. Back to `page2.com`
3. Visit `page4.com`

Expected Output: Forward stack must be empty.
Pass/Fail Criteria: forwardStack.top = = -1

```
╔═══════════════════════════════════╗
║              BROWSER              ║
╠═══════════════════════════════════╣
║ Current Page: google.com          ║
╠═══════════════════════════════════╣
║ 1. Visit new page                 ║
║ 2. Go back                        ║
║ 3. Go forward                     ║
║ 4. Show browser history           ║
║ 5. Exit                           ║
╚═══════════════════════════════════╝
Choose Option: 1
Enter url: page1.com

Visited: page1.com

╔═══════════════════════════════════╗
║              BROWSER              ║
╠═══════════════════════════════════╣
║ Current Page: page1.com           ║
╠═══════════════════════════════════╣
║ 1. Visit new page                 ║
║ 2. Go back                        ║
║ 3. Go forward                     ║
║ 4. Show browser history           ║
```

```
5. Exit
```

```
Choose Option: 1
Enter url: page2.com

Visited: page2.com
```

```
BROWSER
Current Page: page2.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page3.com

Visited: page3.com
```

```
BROWSER
Current Page: page3.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 2

Went back to: page1.com
```

```
BROWSER
```

```
Current Page: page1.com
```

```
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 3
Enter Steps: 1

Went forward to: page2.com
```

```
                BROWSER
```

```
Current Page: page2.com
```

```
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 4

======== Browser Stack View ========
-2. google.com
-1. page1.com
0. page2.com  ⟵ current page
1. page3.com
====================================
```

```
                BROWSER
```

```
Current Page: page2.com
```

```
1. Visit new page
2. Go back
3. Go forward
```

```
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 2

Went back to: google.com
```

```
BROWSER
Current Page: google.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page1.com

Visited: page1.com
```

```
BROWSER
Current Page: page1.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page2.com

Visited: page2.com
```

```
BROWSER
```

```
Current Page: page2.com

1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page3.com

Visited: page3.com
```

```
                BROWSER

Current Page: page3.com

1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 1

Went back to: page2.com
```

```
                BROWSER

Current Page: page2.com

1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 4
```

```
======== Browser Stack View ========
-2. google.com
-1. page1.com
0. page2.com  ⟵ current page
1. page3.com
====================================


╔═══════════════════════════════════════════╗
║                  BROWSER                  ║
╠═══════════════════════════════════════════╣
║ Current Page: page2.com                   ║
╠═══════════════════════════════════════════╣
║ 1. Visit new page                         ║
║ 2. Go back                                ║
║ 3. Go forward                             ║
║ 4. Show browser history                   ║
║ 5. Exit                                   ║
╚═══════════════════════════════════════════╝

Choose Option: 1
Enter url: page4.com

Visited: page4.com


╔═══════════════════════════════════════════╗
║                  BROWSER                  ║
╠═══════════════════════════════════════════╣
║ Current Page: page4.com                   ║
╠═══════════════════════════════════════════╣
║ 1. Visit new page                         ║
║ 2. Go back                                ║
║ 3. Go forward                             ║
║ 4. Show browser history                   ║
║ 5. Exit                                   ║
╚═══════════════════════════════════════════╝

Choose Option: 4

======== Browser Stack View ========
-3. google.com
-2. page1.com
-1. page2.com
0. page4.com  ⟵ current page
```

==========

Test 05 - Show history Formatting

Description: Verifies negative indexes for back pages and positive for forward.
Items Covered: showHistory()
Procedure:

1. Visit four pages.
2. Go back 2 steps.
3. Call show history.

Expected Output

- Back pages shown as: −1, −2, ...
- Current page shown as index 0
- Forward pages shown as +1, +2, ...

Pass/Fail Criteria: Correct structure and order.

```
┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: google.com             │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
│ 2. Go back                           │
│ 3. Go forward                        │
│ 4. Show browser history              │
│ 5. Exit                              │
└──────────────────────────────────────┘
Choose Option: 1
Enter url: page1.com

Visited: page1.com

┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: page1.com              │
│                                      │
```

```
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page2.com

Visited: page2.com
```

```
                BROWSER
Current Page: page2.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page3.com

Visited: page3.com
```

```
                BROWSER
Current Page: page3.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 1

Went back to: page2.com
```

```
┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: page2.com              │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
│ 2. Go back                           │
│ 3. Go forward                        │
│ 4. Show browser history              │
│ 5. Exit                              │
└──────────────────────────────────────┘
Choose Option: 4

======== Browser Stack View ========
-2. google.com
-1. page1.com
0. page2.com  ⟵ current page
1. page3.com
====================================


┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: page2.com              │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
│ 2. Go back                           │
│ 3. Go forward                        │
│ 4. Show browser history              │
│ 5. Exit                              │
└──────────────────────────────────────┘
Choose Option: 1
Enter url: page4.com

Visited: page4.com

┌──────────────────────────────────────┐
│               BROWSER                │
├──────────────────────────────────────┤
│ Current Page: page4.com              │
├──────────────────────────────────────┤
│ 1. Visit new page                    │
```

```
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 4

======== Browser Stack View ========
-3. google.com
-2. page1.com
-1. page2.com
0. page4.com  ⟵ current page
====================================
```

```
BROWSER
Current Page: page4.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: 3

Visited: 3
```

```
BROWSER
Current Page: 3
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 2
```

```
Enter Steps: 4

Went back to: google.com
```

```
╔═══════════════════════════════════════╗
║                BROWSER                ║
╠═══════════════════════════════════════╣
║ Current Page: google.com              ║
╠═══════════════════════════════════════╣
║ 1. Visit new page                     ║
║ 2. Go back                            ║
║ 3. Go forward                         ║
║ 4. Show browser history               ║
║ 5. Exit                               ║
╚═══════════════════════════════════════╝
```

```
Choose Option: 1
Enter url: page1.com

Visited: page1.com
```

```
╔═══════════════════════════════════════╗
║                BROWSER                ║
╠═══════════════════════════════════════╣
║ Current Page: page1.com               ║
╠═══════════════════════════════════════╣
║ 1. Visit new page                     ║
║ 2. Go back                            ║
║ 3. Go forward                         ║
║ 4. Show browser history               ║
║ 5. Exit                               ║
╚═══════════════════════════════════════╝
```

```
Choose Option: 1
Enter url: page2.com

Visited: page2.com
```

```
╔═══════════════════════════════════════╗
║                BROWSER                ║
╠═══════════════════════════════════════╣
║ Current Page: page2.com               ║
╠═══════════════════════════════════════╣
║ 1. Visit new page                     ║
```

```
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page3.com

Visited: page3.com
```

```
                BROWSER
Current Page: page3.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 1
Enter url: page4.com

Visited: page4.com
```

```
                BROWSER
Current Page: page4.com
1. Visit new page
2. Go back
3. Go forward
4. Show browser history
5. Exit
```

```
Choose Option: 2
Enter Steps: 2

Went back to: page2.com
```

```
╔═══════════════════════════════════════╗
║                BROWSER                ║
╠═══════════════════════════════════════╣
║ Current Page: page2.com               ║
╠═══════════════════════════════════════╣
║ 1. Visit new page                     ║
║ 2. Go back                            ║
║ 3. Go forward                         ║
║ 4. Show browser history               ║
║ 5. Exit                               ║
╚═══════════════════════════════════════╝
Choose Option: 4

======== Browser Stack View ========
-2. google.com
-1. page1.com
0. page2.com  ⟵ current page
1. page3.com
2. page4.com
====================================
```

Conclusion

The Browser History Navigation System successfully demonstrates how fundamental data structures—in this case, two stacks—can be used to replicate real-world browser behavior. By separating past and future navigation into a Back Stack and a Forward Stack, the implementation accurately models the operations of visiting new pages, moving backward through history, and moving forward after undoing navigation.

All core operations, including visit(), back(), and forward(), are implemented using simple and efficient stack operations that run in constant time. The system also correctly handles edge cases such as invalid step inputs, duplicate visits, and the clearing of forward history after a new page is loaded. The provided history display function further helps visualize how the navigation state evolves during execution.

Overall, the project demonstrates a clear understanding of stack-based navigation logic and effectively translates theoretical concepts into a practical, interactive C program. The system is reliable, easy to use, and provides an accurate representation of how modern browsers internally manage navigation history.

References

1. Kernighan, B. W., & Ritchie, D. M. The C Programming Language. 2nd ed., Prentice Hall, 1988.
— Used as a reference for C syntax, string handling, pointers, and procedural programming models.

2. Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. Data Structures and Algorithms in C. Wiley, 2016.
— Referenced for stack structures, LIFO principles, and time complexity analysis used in Table 2.

3. Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed., Addison-Wesley, 2004.
— Basis for UML conventions used in design diagrams ("UML diagrams follow standard conventions described by Fowler…").

4. Fagan, M. E. "Design and Code Inspections to Reduce Errors in Program Development." IBM Systems Journal, vol. 15, no. 3, 1976, pp. 182–211.
— Referenced for inspection methodology ("similar to the Fagan Inspection process").

5. Mozilla Developer Network (MDN). "History API."
Available at: https://developer.mozilla.org/en-US/docs/Web/API/History
— Used when describing multi-step navigation inspired by history.back(), history.forward(), and history.go(n).

6. W3C. HTML5 Browser History and Navigation Concepts.
Available at: https://www.w3.org/TR/html52/browsers.html
— Referenced for understanding browser navigation behavior (forward-stack clearing, navigation rules).

7. Chromium Project Documentation. "Navigation and History Handling in Chromium."
Available at: https://chromium.googlesource.com/

— Used to understand real browser internal behavior mentioned in the improvements section.

8. IEEE. IEEE Std 829 – Standard for Software Testing Documentation. IEEE, 2008.
— Referenced implicitly for structuring test cases, pass/fail criteria, and test specification style.

9. Sommerville, I. Software Engineering. 10th ed., Pearson, 2016.
— Indirect reference for software lifecycle concepts, module separation, and recommendations in the conclusion section.