Tutorial 3: Linked Lists

ELEC 278: Fundamentals of Information Structures

The learning goals for Tutorial 3 are:

- Practice linked list operations by building a browser history object.

Problem 1. Replicated from `https://leetcode.com/problems/design-browser-history/`

You have a browser of one tab where you start on the homepage and you can visit another `url`, get back in the history number of steps or move forward in the history number of steps.

Complete the following code to manage the `BrowserHistory` structure.

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

typedef struct Node {
    char url[21]; // assume that any url will only be 21 characters long
    struct Node* prev;
    struct Node* next;
} Node;

typedef struct {
    // to do
} BrowserHistory;

// Function to create a new node
Node* createNode(const char* url) {
    // to do
}

// Function to initialize the BrowserHistory object
BrowserHistory* browserHistoryCreate(char* homepage) {
    // to do: Initializes the BrowserHistory object with the homepage of the
    browser.
}

// Function to visit a new URL
void browserHistoryVisit(BrowserHistory* obj, char* url) {
    // to do: Visits url from the current page. It clears up all the forward
    history.
}

// Function to move back in history
char* browserHistoryBack(BrowserHistory* obj, int steps) {
    // to do: Move steps back in history. If you can only return x steps in the
    history and steps > x, you will return only x steps.
```

```
    //Return the current url after moving back in history at most steps.
}

// Function to move forward in history
char* browserHistoryForward(BrowserHistory* obj, int steps) {
    // to do: Move steps forward in history. If you can only forward x steps in
    the history and steps > x, you will forward only x steps.

    //Return the current url after forwarding in history at most steps.
}

// Function to free the BrowserHistory object
void browserHistoryFree(BrowserHistory* obj) {
    // to do
}

// Example usage
int main() {
    BrowserHistory* browserHistory = browserHistoryCreate("google.com");
    browserHistoryVisit(browserHistory, "leetcode.com");
    browserHistoryVisit(browserHistory, "facebook.com");
    browserHistoryVisit(browserHistory, "youtube.com");
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 1)); //
    facebook.com
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 1)); //
    leetcode.com
    printf("Current URL: %s\n", browserHistoryForward(browserHistory, 1)); //
    facebook.com
    browserHistoryVisit(browserHistory, "linkedin.com");
    printf("Current URL: %s\n", browserHistoryForward(browserHistory, 2)); //
    linkedin.com
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 2)); //
    leetcode.com
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 7)); // google.
    com

    browserHistoryFree(browserHistory);
    return 0;
}
```

Solution.

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

typedef struct Node {
    char url[21]; // assume that any url will only be 21 characters long
    struct Node* prev;
    struct Node* next;
} Node;

typedef struct {
    Node* current;
} BrowserHistory;

// Function to create a new node
Node* createNode(const char* url) {
    Node* newNode = (Node*)malloc(sizeof(Node));
```

```
    strcpy(newNode->url, url);
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Function to initialize the BrowserHistory object
// Initializes the object with the homepage of the browser.
BrowserHistory* browserHistoryCreate(char* homepage) {
    BrowserHistory* obj = (BrowserHistory*)malloc(sizeof(BrowserHistory));
    obj->current = createNode(homepage);
    return obj;
}

// Function to visit a new URL
// Visits url from the current page. It clears up all the forward history.
void browserHistoryVisit(BrowserHistory* obj, char* url) {
    Node* temp;
    while (obj->current->next != NULL) {
        temp = obj->current->next;
        obj->current->next = temp->next;
        free(temp);
    }

    Node* newNode = createNode(url);
    obj->current->next = newNode;
    newNode->prev = obj->current;
    obj->current = newNode;
}

// Function to move back in history
// Move steps back in history. If you can only return x steps in the history and
    steps > x, you will return only x steps. Return the current url after moving
    back in history at most steps.
char* browserHistoryBack(BrowserHistory* obj, int steps) {
    while (steps > 0 && obj->current->prev != NULL) {
        obj->current = obj->current->prev;
        steps--;
    }
    return obj->current->url;
}

// Function to move forward in history
// Move steps forward in history. If you can only forward x steps in the history
    and steps > x, you will forward only x steps. Return the current url after
    forwarding in history at most steps.
char* browserHistoryForward(BrowserHistory* obj, int steps) {
    while (steps > 0 && obj->current->next != NULL) {
        obj->current = obj->current->next;
        steps--;
    }
    return obj->current->url;
}

// Function to free the BrowserHistory object
void browserHistoryFree(BrowserHistory* obj) {
    Node* temp;
    while (obj->current->prev != NULL) {
        obj->current = obj->current->prev;
```

```
    }
    while (obj->current != NULL) {
        temp = obj->current;
        obj->current = obj->current->next;
        free(temp);
    }
    free(obj);
}

// Example usage
int main() {
    BrowserHistory* browserHistory = browserHistoryCreate("google.com");
    browserHistoryVisit(browserHistory, "leetcode.com");
    browserHistoryVisit(browserHistory, "facebook.com");
    browserHistoryVisit(browserHistory, "youtube.com");
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 1)); //
    facebook.com
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 1)); //
    leetcode.com
    printf("Current URL: %s\n", browserHistoryForward(browserHistory, 1)); //
    facebook.com
    browserHistoryVisit(browserHistory, "linkedin.com");
    printf("Current URL: %s\n", browserHistoryForward(browserHistory, 2)); //
    linkedin.com
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 2)); //
    leetcode.com
    printf("Current URL: %s\n", browserHistoryBack(browserHistory, 7)); // google.
    com

    browserHistoryFree(browserHistory);
    return 0;
}
```