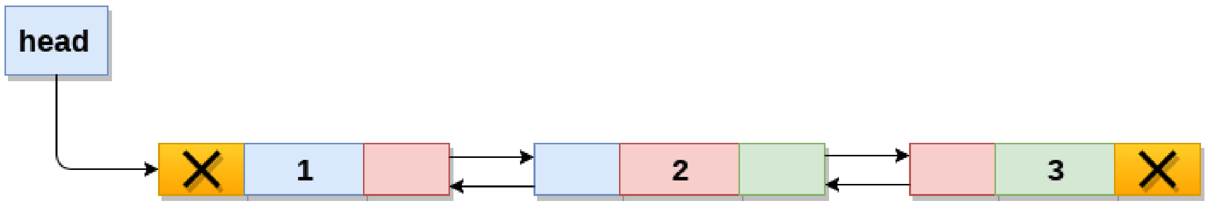
Name	MAUREEN MIRANDA
UID no.	2022300060
Experiment No.	4

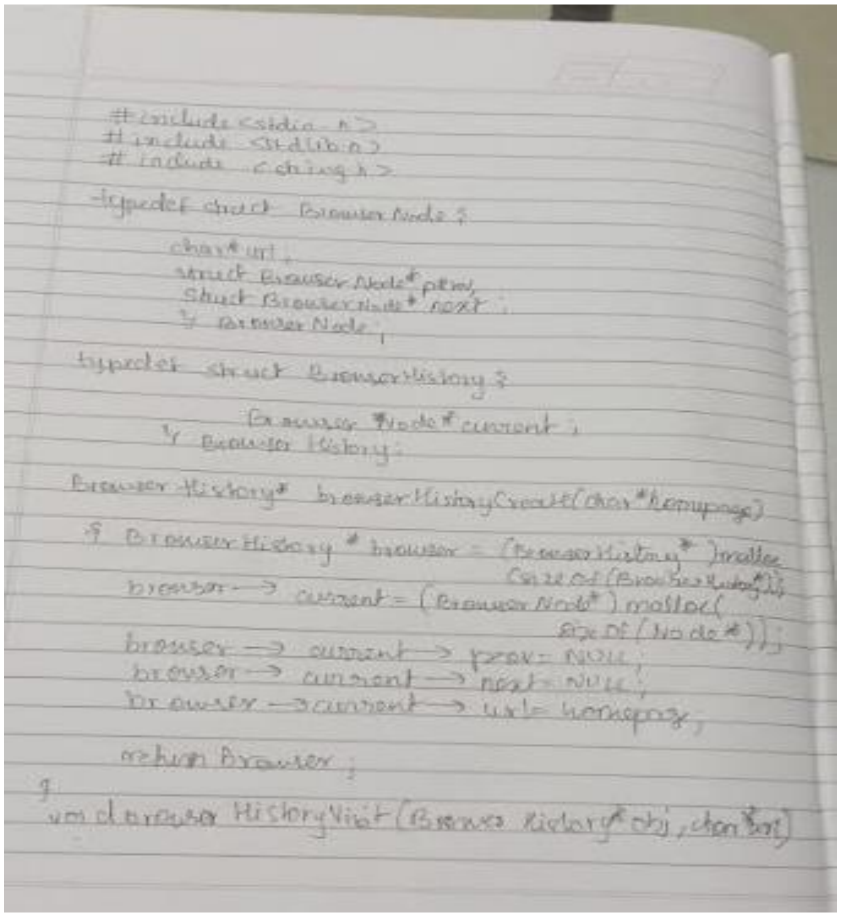
AIM:	TO IMPLEMENT A DOUBLY LINKED LIST AND USE IT
Program 1	
PROBLEM STATEMENT :	Design Browser History
THEORY:	Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence. Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer) , pointer to the previous node (previous pointer). A sample node in a doubly linked list is shown in the figure. Doubly linked list A doubly linked list containing three nodes having numbers from 1 to 3 in their data part, is shown in the following image.  Doubly Linked List In a singly linked list, we could traverse only in one direction, because each node contains address of the next node and it doesn't have any record of its previous nodes. However, doubly linked list overcome this limitation of singly linked list. Due to the fact that, each node of the list contains the address of its previous node, we can find all the details about the previous node as well by using the previous address stored inside the previous part of each node.

ALGORITHM:
An ADT with the name of BrowserHistory has been declared with the following fields:

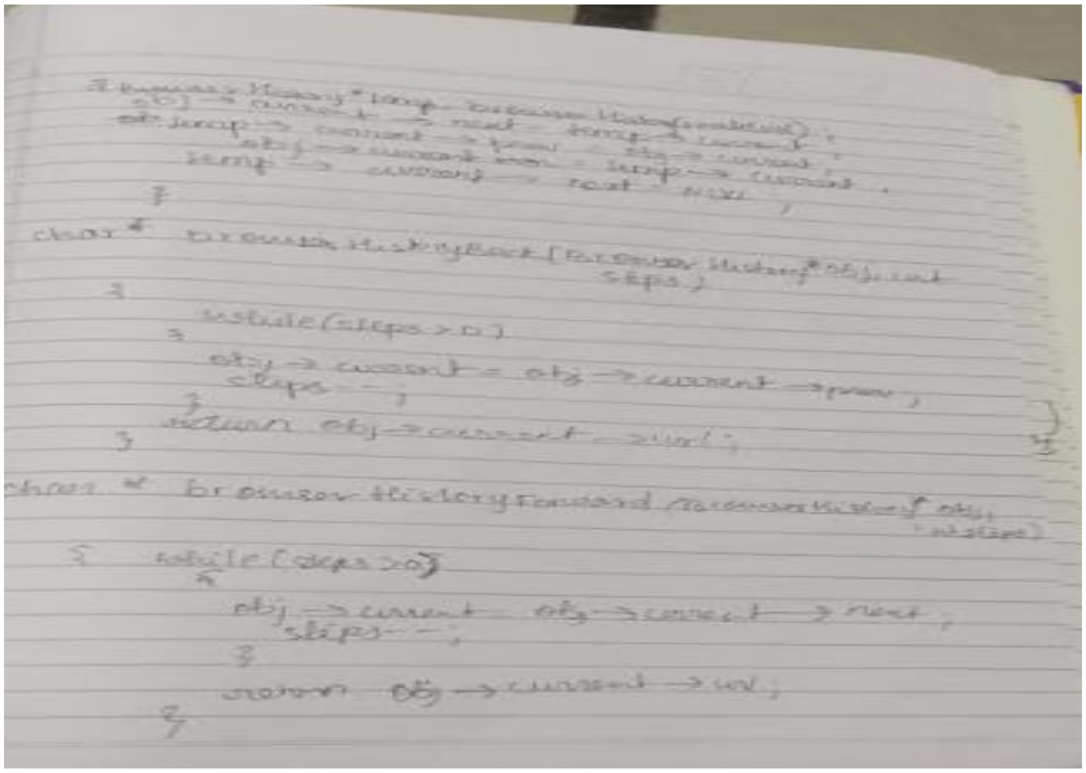
```
Begin procedure browserHistoryCreate(char  homepage):-
Step 1: SET browser := ( BrowserHistory * ) malloc ( sizeof ( BrowserHistory ) )
Step 2: SET browser->current := ( BrowserNode * ) malloc ( sizeof ( BrowserNode ) )
Step 3: SET browser->current := prev = NULL
Step 4: SET browser->current := next = NULL
Step 5: SET browser->current := url = homepage
Step 6: RETURN browser
End procedure


Begin procedure browserHistoryVisit(BrowserHistory  obj , char  url):-
Step 1: SET temp := browserHistoryCreate ( url )
Step 2: SET obj->current := next = temp -> current
Step 3: SET temp->current := prev = obj -> current
Step 4: SET obj->current := temp -> current
Step 5: SET temp->current := next = NULL
Step 6: PRINT url
End procedure


Begin procedure browserHistoryBack(BrowserHistory  obj , int steps):-
Step 1: SET x := steps
Step 2: Repeat step 1 to  2 while steps > 0 and obj -> current -> prev ! = NULL
          Step 1: SET obj->current := obj -> current -> prev
          Step 2: SET steps := steps - 1
End of while block
Step 2: PRINT x , obj -> current -> url
Step 3: RETURN obj -> current -> url
End procedure


Begin procedure browserHistoryForward(BrowserHistory  obj , int steps):-
Step 1: SET x := steps
Step 2: Repeat step 1 to  2 while steps > 0 and obj -> current -> next ! = NULL
```

```
        Step 1: SET obj->current := obj -> current -> next
        Step 2: SET steps := steps - 1
End of while block
Step 2: PRINT x , obj -> current -> url
Step 3: RETURN obj -> current -> url
End procedure


Begin procedure browserHistoryFree(BrowserHistory  obj):-
End procedure
```

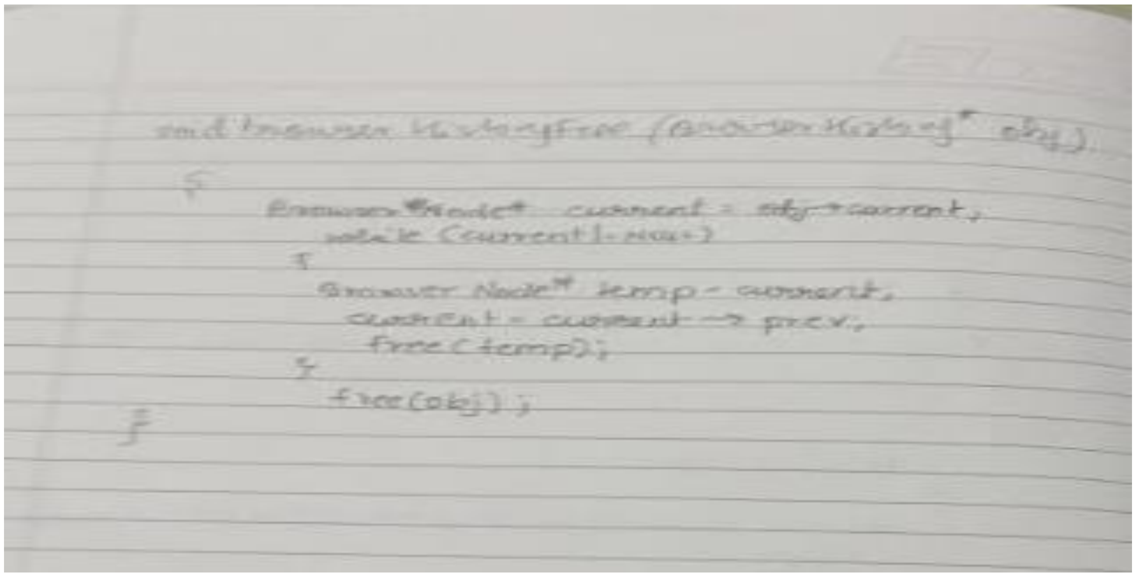
SOLUTION:



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct BrowserNode {
        char* url;
        struct BrowserNode* prev;
        Struct BrowserNode* next;
        } BrowserNode;
typedef struct BrowserHistory {
                BrowserNode* current;
        } BrowserHistory;
BrowserHistory* browserHistoryCreate(char*homepage)
{ BrowserHistory* browser = (BrowserHistory* )malloc
                                        (sizeof(BrowserHistory*));
    browser -> current = (BrowserNode* ) malloc(
                                        sizeof(Node*));
    browser -> current -> prev = NULL;
    browser -> current -> next = NULL;
    browser -> current -> url = homepage;

    return Browser;
}
void browserHistoryVisit(BrowserHistory* obj, char*url)
```



```
 BrowserHistory* temp = browserHistory(url);
  obj -> current -> next = temp -> current ;
 temp -> current -> prev = obj -> current ;
      obj -> current = temp -> current ;
  temp -> current -> next = NULL ;
  }
char* browserHistoryBack(BrowserHistory* obj, int
                          steps)
 {
     while(steps > 0)
    {
     obj -> current = obj -> current -> prev ;
      steps--;
    }
   return obj -> current -> url;
 }
                                                        }
char* browserHistoryForward(BrowserHistory* obj,
                                                  int steps)
 {   while(steps > 0)
       {
        obj -> current = obj -> current -> next ;
         steps--;
        }
        return obj -> current -> url;
      }
```



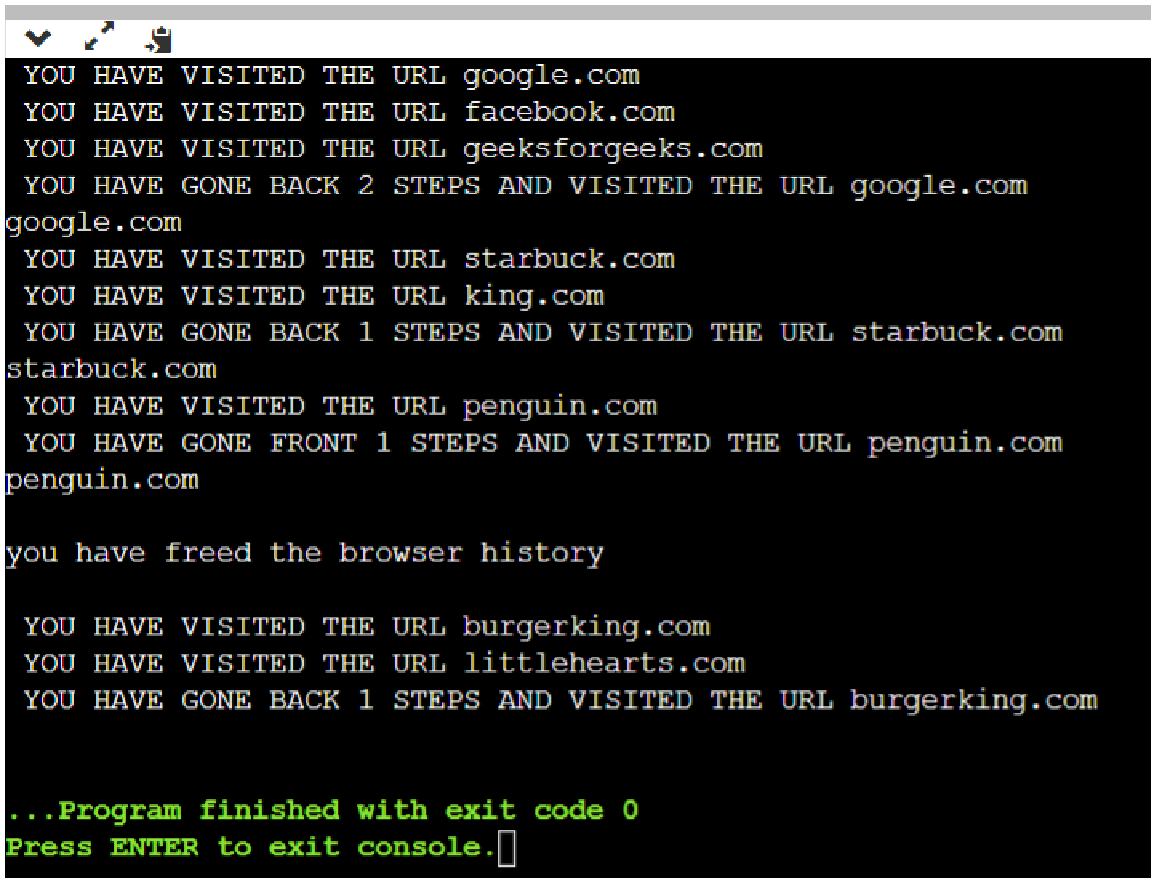
```
 void browserHistoryFree(BrowserHistory* obj)
  {
        BrowserNode* current = obj -> current;
          while (current != NULL)
        {
          BrowserNode* temp = current;
           current = current -> prev;
            free(temp);
        }
          free(obj);
 }
```

PROGRAM:

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

typedef struct BrowserNode{
    char* url;
    // Pointer to the previous node
    struct BrowserNode* prev;
    // Pointer to the next node
    struct BrowserNode* next;
} BrowserNode;


typedef struct BrowserHistory{
    // Pointer to the current node
    BrowserNode* current;
} BrowserHistory;


// this creates a browser history
BrowserHistory* browserHistoryCreate(char * homepage) {
    BrowserHistory * browser = (BrowserHistory*)malloc(sizeof(BrowserHistory));
     browser->current = (BrowserNode*)malloc(sizeof(BrowserNode));
    //browser->current= browser;
    browser->current->prev=NULL;
    browser->current->next=NULL;
    browser->current->url=homepage;

    return browser;
}

//Visits url from the current page. It clears up all the forward history.

// this visits a URL
void browserHistoryVisit(BrowserHistory* obj, char * url)
{
    BrowserHistory*temp=browserHistoryCreate(url);
  obj->current->next=temp->current;
  temp->current->prev=obj->current;
      obj->current=temp->current;

  temp->current->next=NULL;

printf(" YOU HAVE VISITED THE URL %s\n",url);

}



// this moves back a number of 'steps' in history
char * browserHistoryBack(BrowserHistory* obj, int steps) {

int x=steps;

  while(steps>0&& obj->current->prev != NULL)
  {

   obj->current=obj->current->prev;

   steps--;

  }
 printf(" YOU HAVE GONE BACK %d STEPS AND VISITED THE URL %s\n",x,obj-
>current->url);

  return obj->current->url;

}

// this moves forward a number of 'steps' in history
char * browserHistoryForward(BrowserHistory* obj, int steps) {

   int x=steps;
  while(steps>0&& obj->current->next != NULL)
  {
  obj->current = obj->current->next;
  steps--;
  }
  printf(" YOU HAVE GONE FRONT %d STEPS AND VISITED THE URL %s\n",x,obj-
>current->url);
  return obj->current->url;
}

// this cleans up the browser history and releases memory
void browserHistoryFree(BrowserHistory* obj) {

      void browserHistoryFree(BrowserHistory* obj) {
    BrowserNode* current = obj->current;
    while (current != NULL) {
        BrowserNode* temp = current;
        current = current->prev;
        free(temp);
    }
    free(obj); // Free the BrowserHistory object
}



}
int main()
{

BrowserHistory*newnode=browserHistoryCreate("leetcode.com");
//newnode->current=newnode;
browserHistoryVisit(newnode, "google.com");
browserHistoryVisit(newnode, "facebook.com");
browserHistoryVisit(newnode, "geeksforgeeks.com");




printf("%s\n",browserHistoryBack(newnode,2));
browserHistoryVisit(newnode,"starbuck.com");
browserHistoryVisit(newnode,"king.com");
printf("%s\n",browserHistoryBack(newnode,1));
browserHistoryVisit(newnode,"penguin.com");
printf("%s\n",browserHistoryForward(newnode,1));
printf("\nyou have freed the browser history\n\n");
browserHistoryFree(newnode);
```

```
browserHistoryVisit(newnode,"burgerking.com");
browserHistoryVisit(newnode,"littlehearts.com");
browserHistoryBack(newnode,1);


return 0;
}
```

RESULT:



```
 YOU HAVE VISITED THE URL google.com
 YOU HAVE VISITED THE URL facebook.com
 YOU HAVE VISITED THE URL geeksforgeeks.com
 YOU HAVE GONE BACK 2 STEPS AND VISITED THE URL google.com
google.com
 YOU HAVE VISITED THE URL starbuck.com
 YOU HAVE VISITED THE URL king.com
 YOU HAVE GONE BACK 1 STEPS AND VISITED THE URL starbuck.com
starbuck.com
 YOU HAVE VISITED THE URL penguin.com
 YOU HAVE GONE FRONT 1 STEPS AND VISITED THE URL penguin.com
penguin.com

you have freed the browser history

 YOU HAVE VISITED THE URL burgerking.com
 YOU HAVE VISITED THE URL littlehearts.com
 YOU HAVE GONE BACK 1 STEPS AND VISITED THE URL burgerking.com


...Program finished with exit code 0
Press ENTER to exit console.
```