Assignment 2 DS

Assignment 2 DS

- c

```c
/*
Assignment 2 - Data Structures
Insert (AtHead,AtLast, AtPos), Delete (AtHead,AtLast, AtPos), Display, Palindrome, Re
*/

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

struct Node
{
    int data;
    struct Node *next;
};

struct Node *create_Node(int data)
{
    struct Node *new = (struct Node *)malloc(sizeof(struct Node));
    // head->next = new;
    new->data = data;
    new->next = NULL;
    return new;
}

void displayLL(struct Node *head)
{
    int i = 0;
    head = head->next;
    while (head != NULL & i < 10)
    {
        i++;
        printf("%dth element: %d \n", i, head->data);
        head = head->next;
    }
    printf("_____________\n");
```

```c
}

void insertAtLast(struct Node *head, int data)
{
    struct Node *new = (struct Node *)malloc(sizeof(struct Node));
    struct Node *temp = head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = new;
    new->data = data;
    new->next = NULL;
    // return new;
}

void insertAtFirst(struct Node *head, int data)
{
    struct Node *new = (struct Node *)malloc(sizeof(struct Node));
    new->next = head->next;
    head->next = new;
    new->data = data;
    // new->next = NULL;
    // return new;
}

void insertAtPos(struct Node *head, int pos, int data)
{
    struct Node *temp = head;
    struct Node *new = create_Node(data);
    int i = 0;
    while (i != 0)
    {
        temp = temp->next;
    }

    new->next = temp->next;
    temp->next = new;
}
void deleteAtFirst(struct Node *head)
{
    struct Node *temp = head->next;
    head->next = head->next->next;
    temp->next = NULL;
}

void deleteAtLast(struct Node *head)
{
    struct Node *temp = head;
    while (temp->next->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = NULL;
}

void deleteAtPos(struct Node *head, int pos)
{
    struct Node *temp = head->next;
    // struct Node* new = create_Node(data);
    int i = 0;
    while (i != pos)
    {
        i++;
        temp = temp->next;
    }
    struct Node *p = temp->next;
    // new->next = temp->next;
    temp->next = p->next;
    p->next = NULL;
}

struct Node *reverse(struct Node *head)
{
    struct Node *prev = NULL;
    struct Node *current = head->next;
    struct Node *next = NULL;

    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }

    head->next = prev;
    displayLL(head);
}

int count(struct Node *head)
{
    int i = 0;
    while (head->next != NULL)
    {
        i++;
        head = head->next;
    }
    return i;
}

int isPalindrome(struct Node *head)
{
    struct Node *slow = head->next;
    struct Node *fast = head->next;
    struct Node *prev_slow = head;
    struct Node *mid = NULL;
    struct Node *second_half = NULL;
    int is_palindrome = 1;

    // Find the middle node of the linked list
    while (fast != NULL && fast->next != NULL)
    {
        fast = fast->next->next;
        prev_slow = slow;
        slow = slow->next;
    }

    // If the length of the linked list is odd, move slow pointer one step ahead
    if (fast != NULL)
    {
        mid = slow;
        slow = slow->next;
    }

    // Reverse the second half of the linked list
```

```c
    second_half = slow;
    prev_slow->next = NULL;
    reverse(second_half);

    // Compare the first half and the reversed second half of the linked list
    struct Node *p1 = head->next;
    struct Node *p2 = second_half;
    while (p1 != NULL && p2 != NULL)
    {
        if (p1->data != p2->data)
        {
            is_palindrome = 0;
            break;
        }
        p1 = p1->next;
        p2 = p2->next;
    }

    // Reverse the second half back to its original order
    reverse(second_half);
    prev_slow->next = mid;
    if (mid != NULL)
    {
        mid->next = second_half;
    }
    else
    {
        prev_slow->next = second_half;
    }

    return is_palindrome;
}

int main()
{
    printf("Starting program! \n");
    struct Node *head = (struct Node *)malloc(sizeof(struct Node));
    // head->data = 0;
    head->next = NULL;
    // struct Node *n1 = create_Node(8);
    // displayLL(head);
    // struct Node *n2 = create_Node(2);
    // displayLL(head);
    // struct Node *n3 = create_Node(5);
    // displayLL(head);

    // insert at last
    insertAtLast(head, 1);
    insertAtLast(head, 2);
    insertAtLast(head, 3);
    insertAtLast(head, 4);
    insertAtLast(head, 5);
    displayLL(head);
    deleteAtPos(head, 1);
    displayLL(head);
    // insert at first
    insertAtFirst(head, 6);
    displayLL(head);
    deleteAtFirst(head);
    displayLL(head);
    deleteAtLast(head);
    displayLL(head);
    // reverse(head);
    printf("Count: %d \n", count(head));
    reverse(head);
    // displayLL(head);
    isPalindrome(head) ? printf("The LinkedList is a palindrome!") : printf("The Link

    return 0;
```

- **Output**
-

```output
Starting program!
1th element: 1
2th element: 2
3th element: 3
4th element: 4
5th element: 5
_____________
1th element: 1
2th element: 2
```

```
3th element: 4
4th element: 5
_____________
1th element: 6
2th element: 1
3th element: 2
4th element: 4
5th element: 5
_____________
1th element: 1
2th element: 2
3th element: 4
4th element: 5
_____________
1th element: 1
2th element: 2
3th element: 4
_____________
Count: 3
1th element: 4
2th element: 2
3th element: 1
_____________
_____________
_____________
The LinkedList is not a palindrome!
```

-

1 Linked Reference

Sep 3rd, 2024

- [[Assignment 2 DS]]

Unlinked References