DAY 21-DAILY ASSIGNMENTS

ANSU MARIUM SHIBU

04-12-2024

1.

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

struct node {
    int data;
    struct node *next;
} *head = NULL;

// Function declarations
void display(struct node *p);
void recdisplay(struct node *p);
int count(struct node *p);
int reccount(struct node *p);
int sum(struct node *p);
int recsum(struct node *p);
int max(struct node *p);
int recmax(struct node *p);
struct node *search(struct node *p, int key);
struct node *insert(struct node *head, int index, int x);

int main() {
    struct node *second, *third, *temp;

    // Allocating memory for nodes
    head = (struct node *)malloc(sizeof(struct node));
    second = (struct node *)malloc(sizeof(struct node));
    third = (struct node *)malloc(sizeof(struct node));

    // Initializing node data and linking them
```

```c
// Initializing node data and linking them
head->data = 10;
head->next = second;

second->data = 20;
second->next = third;

third->data = 30;
third->next = NULL;

printf("Original List:\n");
recdisplay(head); // Recursive display

// Count nodes
int ncount = count(head);
printf("Total number of nodes (non-recursive): %d\n", ncount);

int rcount = reccount(head);
printf("Total number of nodes (recursive): %d\n", rcount);

// Sum of elements
int nsum = sum(head);
printf("Sum of elements (non-recursive): %d\n", nsum);

int rsum = recsum(head);
printf("Sum of elements (recursive): %d\n", rsum);

// Maximum element
```

```c
    int rsum = recsum(head);
    printf("Sum of elements (recursive): %d\n", rsum);

    // Maximum element
    int nmax = max(head);
    printf("Maximum element (non-recursive): %d\n", nmax);

    int rmax = recmax(head);
    printf("Maximum element (recursive): %d\n", rmax);

    // Search for a key
    temp = search(head, 20); // Search for the key 20
    if (temp != NULL) {
        printf("Element found: %d\n", temp->data);
    } else {
        printf("Element not found.\n");
    }

    // Insertions
    head = insert(head, 0, 5);   // Insert 5 at the beginning
    head = insert(head, 2, 15);  // Insert 15 at index 2
    head = insert(head, 5, 35);  // Insert 35 at the end

    printf("Updated List:\n");
    recdisplay(head); // Display the updated list

    return 0;
```

```c
    printf("Updated List:\n");
    recdisplay(head); // Display the updated list

    return 0;
}

// Function to display the list iteratively
void display(struct node *p) {
    while (p != NULL) {
        printf("%d -> ", p->data);
        p = p->next;
    }
    printf("NULL\n");
}

// Function to display the list recursively
void recdisplay(struct node *p) {
    if (p != NULL) {
        printf("%d -> ", p->data);
        recdisplay(p->next);
    } else {
        printf("NULL\n");
    }
}

// Function to count nodes iteratively
int count(struct node *p) {
    int c = 0;
```

```c
// Function to count nodes iteratively
int count(struct node *p) {
    int c = 0;
    while (p != NULL) {
        c++;
        p = p->next;
    }
    return c;
}

// Function to count nodes recursively
int reccount(struct node *p) {
    if (p == NULL) {
        return 0;
    } else {
        return reccount(p->next) + 1;
    }
}

// Function to calculate the sum of elements iteratively
int sum(struct node *p) {
    int total = 0;
    while (p != NULL) {
        total += p->data;
        p = p->next;
    }
    return total;
```

```
        total += p->data;
        p = p->next;
    }
    return total;
}

// Function to calculate the sum of elements recursively
int recsum(struct node *p) {
    if (p == NULL) {
        return 0;
    } else {
        return p->data + recsum(p->next);
    }
}

// Function to find the maximum element iteratively
int max(struct node *p) {
    int maxi = -32768; // Assuming small value as initial max
    while (p != NULL) {
        if (p->data > maxi) {
            maxi = p->data;
        }
        p = p->next;
    }
    return maxi;
}

// Function to find the maximum element recursively
```

```
int max(struct node *p) {
}

// Function to find the maximum element recursively
int recmax(struct node *p) {
    if (p == NULL) {
        return -32768; // Base case for empty list
    }
    int x = recmax(p->next);
    return (p->data > x) ? p->data : x;
}

// Function to search for a key in the list
struct node *search(struct node *p, int key) {
    while (p != NULL) {
        if (key == p->data) {
            return p;
        }
        p = p->next;
    }
    return NULL; // Key not found
}

// Function to insert a node at a specific index
struct node *insert(struct node *head, int index, int x) {
    struct node *t, *p = head;
    int i;

    // Validate the index
    if (index < 0 || index > count(p)) {
```

```c

// Function to insert a node at a specific index
struct node *insert(struct node *head, int index, int x) {
    struct node *t, *p = head;
    int i;

    // Validate the index
    if (index < 0 || index > count(p)) {
        printf("Invalid index\n");
        return head;
    }

    // Create a new node
    t = (struct node *)malloc(sizeof(struct node));
    t->data = x;

    if (index == 0) { // Insertion at the beginning
        t->next = head;
        head = t;
    } else { // Insertion at other positions
        for (i = 0; i < index - 1; i++) {
            p = p->next;
        }
        t->next = p->next;
        p->next = t;
    }

    return head;
```

```
PS D:\c progrms coding> gcc linkedlist1.c
PS D:\c progrms coding> ./a
Original List:
10 -> 20 -> 30 -> NULL
Total number of nodes (non-recursive): 3
Total number of nodes (recursive): 3
Sum of elements (non-recursive): 60
Sum of elements (recursive): 60
Maximum element (non-recursive): 30
Maximum element (recursive): 30
Element found: 20
Updated List:
5 -> 10 -> 15 -> 20 -> 30 -> 35 -> NULL
PS D:\c progrms coding>
```