DAY 16-DAILY ASSIGNMENTS

27-11-2024

ANSU MARIUM SHIBU

1.C reate a node in a linked list which will have the following details of student 1. Name, roll number, class, section, an array having marks of any three subjects Create a liked for 5 students and print it.

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

typedef struct student {
    char name[50];
    int rollno;
    char class[10];
    char section[10];
    int marks[3];
} student;

typedef struct node {
    student students;
    struct node *next;
} node;

int main() {
    node *head = NULL, *temp, *newnode;

    for (int i = 0; i < 5; i++) {
        newnode = (node*)malloc(sizeof(node));

        printf("Enter details for student %d\n", i + 1);

        printf("Name: ");
```

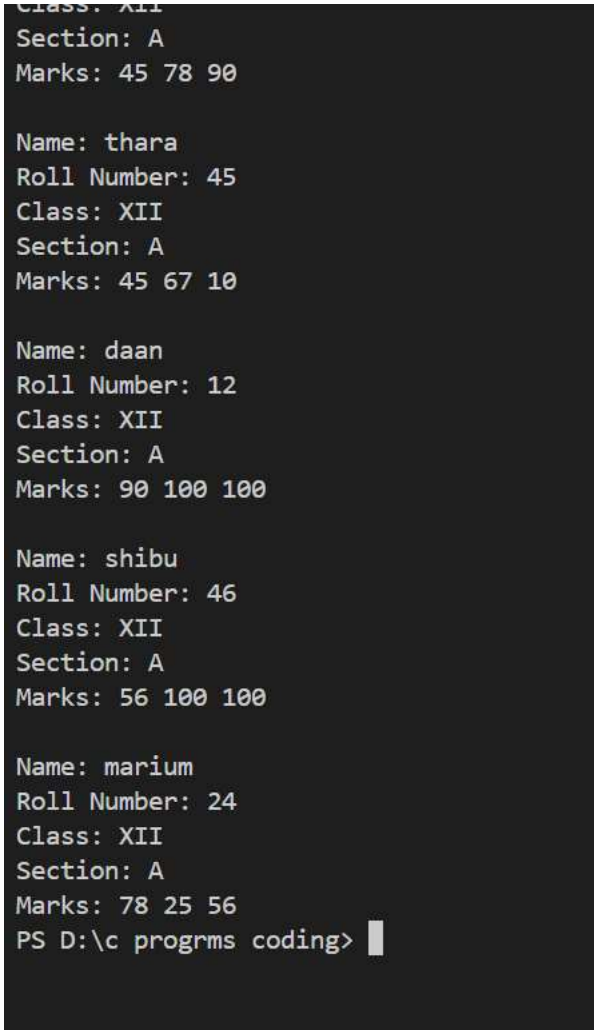
```
    node *head = NULL, *temp, *newnode;

    for (int i = 0; i < 5; i++) {
        newnode = (node*)malloc(sizeof(node));

        printf("Enter details for student %d\n", i + 1);

        printf("Name: ");
        scanf(" %[^\n]", newnode->students.name);

        printf("Roll Number: ");
        scanf("%d", &newnode->students.rollno);

        printf("Class: ");
        scanf("%s", newnode->students.class);

        printf("Section: ");
        scanf("%s", newnode->students.section);

        printf("Enter marks for 3 subjects: ");
        for (int j = 0; j < 3; j++) {
            scanf("%d", &newnode->students.marks[j]);
        }
```

```
    scanf("%s", newnode->students.section);

    printf("Enter marks for 3 subjects: ");
    for (int j = 0; j < 3; j++) {
        scanf("%d", &newnode->students.marks[j]);
    }

    newnode->next = NULL;

    if (head == NULL) {
        head = newnode;
    } else {
        temp->next = newnode;
    }

    temp = newnode;
}

printf("\nStudent Details:\n");
temp = head;
while (temp != NULL) {
    printf("\nName: %s\n", temp->students.name);
    printf("Roll Number: %d\n", temp->students.rollno);
```

```
    for (int i = 0; i < 5; i++) {
        temp = newNode;
    }

    printf("\nStudent Details:\n");
    temp = head;
    while (temp != NULL) {
        printf("\nName: %s\n", temp->students.name);
        printf("Roll Number: %d\n", temp->students.rollno);
        printf("Class: %s\n", temp->students.class);
        printf("Section: %s\n", temp->students.section);
        printf("Marks: ");
        for (int i = 0; i < 3; i++) {
            printf("%d ", temp->students.marks[i]);
        }
        printf("\n");

        temp = temp->next;
    }

    return 0;
}
```

```
PS D:\c progrms coding> gcc singlelink1.c
PS D:\c progrms coding> ./a
Enter details for student 1
Name: ansu
Roll Number: 23
Class: XII
Section: A
Enter marks for 3 subjects: 45
78
90
Enter details for student 2
Name: thara
Roll Number: 45
Class: XII
Section: A
Enter marks for 3 subjects: 45 67 10
Enter details for student 3
Name: daan
Roll Number: 12
Class: XII
Section: A
Enter marks for 3 subjects: 90 100 100
Enter details for student 4
Name: shibu
Roll Number: 46
Class: XII
Section: A
Enter marks for 3 subjects: 56
100
100
Enter details for student 5
Name: marium
```

```
Name: mariam
Roll Number: 24
Class: XII
Section: A
Enter marks for 3 subjects: 78
25
56

Student Details:

Name: ansu
Roll Number: 23
Class: XII
Section: A
Marks: 45 78 90

Name: thara
Roll Number: 45
Class: XII
Section: A
Marks: 45 67 10

Name: daan
Roll Number: 12
Class: XII
Section: A
Marks: 90 100 100

Name: shibu
Roll Number: 46
Class: XII
Section: A
```



2. Problem 1: Reverse a Linked List

Write a C program to reverse a singly linked list. The program should traverse the list, reverse the pointers between the nodes, and display the reversed list.

Requirements:

Define a function to reverse the linked list iteratively.

Update the head pointer to the new first node.

Display the reversed list.

Example Input:

rust

Copy code

Initial list: 10 -> 20 -> 30 -> 40

Example Output:

rust

Copy code

Reversed list: 40 -> 30 -> 20 -> 10

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

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

void InsertEnd(Node** head, int data) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = data;
    new_node->next = NULL;

    if (*head == NULL) {
        *head = new_node;
        return;
    }

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

void printList(Node* head) {
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
```

```c
void printList(Node* head) {
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}

void reverseList(Node** head) {
    Node* prev = NULL;
    Node* current = *head;
    Node* next = NULL;

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

int main() {
    Node* head = NULL;

    InsertEnd(&head, 10);
    InsertEnd(&head, 20);
    InsertEnd(&head, 30);
```

```
void reverseList(Node** head) {
    while (current != NULL) {
        prev = current;
        current = next;
    }
    *head = prev;
}

int main() {
    Node* head = NULL;

    InsertEnd(&head, 10);
    InsertEnd(&head, 20);
    InsertEnd(&head, 30);
    InsertEnd(&head, 40);

    printf("Initial list: ");
    printList(head);

    reverseList(&head);

    printf("Reversed list: ");
    printList(head);

    return 0;
}
```

```
PS D:\c progrms coding> gcc singlelinkass11.c
PS D:\c progrms coding> ./a
Initial list: 10 -> 20 -> 30 -> 40 -> NULL
Reversed list: 40 -> 30 -> 20 -> 10 -> NULLPS
```

3. Problem 2: Find the Middle Node

Write a C program to find and display the middle node of a singly linked list. If the list has an even number of nodes, display the first middle node.

Requirements:

Use two pointers: one moving one step and the other moving two steps.

When the faster pointer reaches the end, the slower pointer will point to the middle node.

Example Input:

rust

Copy code

List: 10 -> 20 -> 30 -> 40 -> 50

Example Output:

scss

Copy code

Middle node: 30

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

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

void InsertEnd(Node** head, int data) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = data;
    new_node->next = NULL;

    if (*head == NULL) {
        *head = new_node;
        return;
    }

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

void printList(Node* head) {
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
```

```
glelinkassi2.c > InsertEnd(Node **, int)
void InsertEnd(Node** head, int data) {
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = new_node;
}

void printList(Node* head) {
    while (head != NULL) {
        printf("%d -> ", head->data);
        head = head->next;
    }
    printf("NULL\n");
}

void findMiddle(Node* head) {
    if (head == NULL) {
        printf("The list is empty.\n");
        return;
    }

    Node* slow = head;
    Node* fast = head;

    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
    }

    printf("Middle node: %d\n", slow->data);
```

```c
    Node* slow = head;
    Node* fast = head;

    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
    }

    printf("Middle node: %d\n", slow->data);
}

int main() {
    Node* head = NULL;

    InsertEnd(&head, 10);
    InsertEnd(&head, 20);
    InsertEnd(&head, 30);
    InsertEnd(&head, 40);
    InsertEnd(&head, 50);

    printf("List: ");
    printList(head);

    findMiddle(head);

    return 0;
}
```

```
PS D:\c progrms coding> gcc singlelinkassi2.c
PS D:\c progrms coding> ./a
List: 10 -> 20 -> 30 -> 40 -> 50 -> NULL
Middle node: 30
PS D:\c progrms coding>
```

4. Problem 3: Detect and Remove a Cycle in a Linked List

Write a C program to detect if a cycle (loop) exists in a singly linked list and remove it if present. Use Floyd's Cycle Detection Algorithm (slow and fast pointers) to detect the cycle.

Requirements:

Detect the cycle in the list.

If a cycle exists, find the starting node of the cycle and break the loop.

Display the updated list.

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

// Define the structure for a node
typedef struct node {
    int data;
    struct node* next;
} Node;

// Function prototypes
void insert(Node** head, int data);
void createCycle(Node* head, int position);
void detectAndRemoveCycle(Node* head);
void printList(Node* head);

int main() {
    Node* head = NULL;

    // Create a linked list
    insert(&head, 10);
    insert(&head, 20);
    insert(&head, 30);
    insert(&head, 40);
    insert(&head, 50);

    // Create a cycle (50 points back to 30)
    createCycle(head, 3);

    // Detect and remove the cycle
```

```
    return 0;
}

// Function to insert a node at the end of the list
void insert(Node** head, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;

    if (*head == NULL) {
        *head = newNode;
        return;
    }

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

// Function to create a cycle in the list
void createCycle(Node* head, int position) {
    Node* temp = head;
    Node* cycleNode = NULL;
    int count = 1;
```

```
// Function to create a cycle in the list
void createCycle(Node* head, int position) {
    Node* temp = head;
    Node* cycleNode = NULL;
    int count = 1;

    while (temp->next != NULL) {
        if (count == position) {
            cycleNode = temp;
        }
        temp = temp->next;
        count++;
    }
    temp->next = cycleNode;  // Create the cycle
}

// Function to detect and remove a cycle
void detectAndRemoveCycle(Node* head) {
    Node* slow = head;
    Node* fast = head;

    // Detect cycle
    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;

        // Cycle detected
```

```
            slow = slow->next;
            fast = fast->next->next;

            // Cycle detected
            if (slow == fast) {
                printf("Cycle detected.\n");

                // Find the start of the cycle
                slow = head;
                while (slow != fast) {
                    slow = slow->next;
                    fast = fast->next;
                }

                // Break the cycle
                Node* temp = fast;
                while (temp->next != slow) {
                    temp = temp->next;
                }
                temp->next = NULL;  // Remove the loop
                printf("Cycle removed.\n");
                return;
            }
        }

        printf("No cycle detected.\n");
    }
```

```
}

// Function to print the list
void printList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```

```
PS D:\c progrms coding> gcc singlelinkassi3.c
PS D:\c progrms coding> ./a
Cycle detected.
Cycle removed.
10 -> 20 -> 30 -> 40 -> 50 -> NULL
PS D:\c progrms coding>
```