CLASS WORKS DAY_16

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

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

}Node;

void insertFront(Node**,int);
void insertMiddle(Node*,int,int);
//function with dual purpose.creating a new node
void insertEnd(Node**,int);
void printList(Node*);
void DelNode(Node**,int);


int main(){

    Node *head=NULL;
    insertEnd(&head,6);//double pointer(stores address of a pointer)
    insertEnd(&head,7);
    insertFront(&head,8);
    insertMiddle(head,2,1);
    insertMiddle(head,3,1);

    DelNode(&head, 8);
    printList(head);
    return 0;
}
// inserting a node in end
void insertEnd(Node** ptrhead,int nData){
    //1.crearing a Node
    Node* newNode=(Node*)malloc(sizeof(Node));
    //1.1 create one more pointer which will point to the last element of the linked list
    Node *ptrTail;
    ptrTail=*ptrhead;
    //the two pointers are pointing to same Node

    //2.enter nData
```

```
    newNode->data=nData;
    //3.make the next fiels as NULL
    newNode->next=NULL;
    //4.if the linked list is empty make ptrhead points to the new Node
    if(*ptrhead==NULL){
        *ptrhead=newNode;
        return;

    }else{
        //5.else traverse list till the last node and insert the new node
    while(ptrTail->next!=NULL){
            //move the ptrtail to the end
            ptrTail=ptrTail->next;

        }
    }
    ptrTail->next=newNode;
}

void printList(Node* node){
    while(node!=NULL){
        printf("%d->",node->data);
        node=node->next;
    }
}
// inserting a node in the begining
void insertFront(Node**ptrhead,int nData){
    // create a new node
    Node* newNode=(Node*)malloc(sizeof(Node));

    // assign data to the new node
    newNode->data=nData;
    // make the new node point to the first node
    newNode->next=(*ptrhead);
    // assign a address of new node to ptrhead
    (*ptrhead)=newNode;

}

// inserting a element any where in the link list

void insertMiddle(Node*ptrhead,int nData,int position){
    Node* newNode=(Node*)malloc(sizeof(Node));
    Node*temp;
```

```
    temp=ptrhead;
    int i=1;
    newNode->data=nData;
    while(temp!=NULL && i<position){
        temp = temp->next;
        i++;

    }
    newNode->next=temp->next;
    temp->next=newNode;

}
void DelNode(Node**head,int key){
    // 10->7->6->1->5->
    // create a temperory node that point to head node
    Node *temp =*head;
    Node *prev=NULL;

    //cas 1:if the key is the first node
    if(temp!=NULL && temp->data==key){
        *head=temp->next; //now head is pointing to 7
        free(temp);

    }
    // case 2:if the key is present somewhere in b/w
    // case3:if the key is not present
}
```

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

// define the structure
typedef struct Node{
    int data;
    struct Node *next;

}Node;

Node* createNode(int data);
```

```
int main(){
//10_>null
   Node*first =createNode(10);

   // 10->20->null
   first->next=createNode(20);
   // 10->20->30
   first->next->next=createNode(30);
   // 10->20->30->40
   first->next->next->next=createNode(40);

   Node *temp;
   temp =first ;


   while (temp!=NULL)
   {
      printf("%d->",temp->data); //10 // 20 // 30
      temp=temp->next; //points to the next address//second//third

   }




}
Node *createNode(int data){
   Node *newNode = (Node *)malloc(sizeof(Node));
   newNode->data=data;
   // initially assigned the next field of newly created node to null
   newNode->next=NULL;
   return newNode;
}
```

ASSIGNMENT

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

```
typedef struct node
{
   char name[50];
   int roll_no;
   int class;
   char section;
   int marks[3];
   struct node *link;
}student;

int main()
{
   student *head = NULL;
   printf("Enter details of 5 students:\n");
   for(int i=0; i<5; i++)
   {
      student *new = (student *)malloc(sizeof(student));
      student *temp;
      printf("\nStudent %d\n", i+1);
      printf("Enter the name of student: ");
      scanf(" %[^\n]", new->name);
      printf("Enter the roll no: ");
      scanf(" %d",&new->roll_no);
      printf("Enter the class and section: ");
      scanf("%d %c", &new->class, &new->section);
      printf("Enter the marks of 3 subjects: \n");
      for(int j=0; j<3; j++)
      {
         printf("Subject %d: ", j+1);
         scanf("%d",&new->marks[j]);
      }
      new->link = NULL;

      if(head == NULL)
      {
         head = new;
      }
      else
      {
         temp=head;

         while(temp->link != NULL)
         {
```

```
            temp = temp->link;
        }
        temp->link = new;
    }

}

//Display details
// traversing

printf("\nStudent details\n");
student *temp ;
temp= head;
int i=1;
while(temp != NULL)
{
    printf("\nStudent %d\n", i);
    printf("Name: %s\n", temp->name);
    printf("Roll no: %d\n", temp->roll_no);
    printf("Class %d, Sec: %c\n", temp->class, temp->section);
    printf("Marks: %d %d %d\n", temp->marks[0], temp->marks[1], temp->marks[2]);
    i++;
    temp = temp->link;
}




return 0;
}



// 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


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

typedef struct Node{
    int data;
    struct Node *next;
}Node;
void read(Node**head,int n);
void display(Node*head);
void reverse(Node**head);

int main(){
    Node*head =NULL;
    int n,data;
    printf("enter the number");
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&data);

     read(&head,data);
    }

    reverse(&head);
    display(head);

}
void read(Node**head,int n){
    Node*new=(Node*)malloc(sizeof(Node));
    Node*temp;
    new->data=n;
    new->next=NULL;
    if(*head==NULL){
        *head=new;
    }
    else{
        temp=*head;
        while(temp->next!=NULL){
```

```
            temp=temp->next;

        }
        temp->next=new;
    }
}

// reverse

void reverse(Node**head){
    Node *prevous;
    Node *current;
    Node *next;
    prevous=NULL;
    current=*head;
    next=NULL;
    while(current!=NULL){
        next=current->next;
        current->next=prevous;
        prevous=current;
        current=next;
    }
    *head=prevous;
}

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

// 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

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


typedef struct Node{
    int data;
    struct Node *next;
}Node;
void add(Node**head,int n);
void middle(Node*head);
void display(Node*head);

int main(){
    Node*head=NULL;
    int n,dat;
    printf("enter the number");
    scanf("%d",&n);
    printf("enter the numbers");
    for(int i=0;i<n;i++){
        scanf("%d",&dat);
add(&head,dat);
    }
    middle(head);

    printf("\n");
    display(head);
}
// traversing
void add(Node**head,int n){
    Node*new = (Node*)malloc(sizeof(Node));
    Node*temp;
    new->data=n;
```

```
    new->next=NULL;
    if(*head==NULL){
        *head=new;
    }
    else{
        temp=*head;
        while(temp->next!=0){
            temp=temp->next;
        }
        temp->next=new;
    }
}
void middle(Node*head){

    Node*first;
    Node*second;
    if(head==NULL){
        printf("this list is empthy");
        return;
    }
    first=head;
    second=head;
    while(second!=NULL && second->next!=NULL){
        first =first->next;
        second=second->next->next;
    }
printf("%d the middile number",first->data);
}
void display(Node*head){
    while(head!=NULL){
        printf("%d->",head->data);
        head=head->next;
    }

}
```

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

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

}Node;

void insertFront(Node**,int);
void insertMiddle(Node*,int);
//function with dual purpose.creating a new node
void insertEnd(Node**,int);
void printList(Node*);

int main(){

   Node *head=NULL;
   insertEnd(&head,6);//double pointer(stores address of a pointer)
   insertEnd(&head,7);
   printList(head);
   return 0;
}
void insertEnd(Node** ptrhead,int nData){
   //1.crearing a Node
   Node* newNode=(Node*)malloc(sizeof(Node));
   //1.1 create one more pointer which will point to the last element of the linked list
   Node *ptrTail;
   ptrTail=*ptrhead;//the two pointers are pointing to same Node

   //2.enter nData
```

```
#include <stdlib.h>

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

void insertLast(Node **head,int data);
void printlist(Node *head);
void createCycle(Node **head,int cycle_index);
void find_cyclic(Node **head);
void remove_cycle(Node **head);

int main(){
   Node *head = NULL;
   int op;
   do{
      printf("\nSingle Linked List operations\n");
      printf("1. Insert at last\n");
      printf("2. Print list\n");
      printf("3. Create cycle (for testing)\n");
      printf("4.Find cyclic linked list\n");
      printf("5. Remove cycle\n");
      printf("6. Exit\n");
      printf("Choose an option: ");
      scanf("%d", &op);

       switch(op)
      {
         case 1:
         {
            // Insert last
            int data;
            printf("Enter the data to be inserted: ");
            scanf("%d", &data);
            insertLast(&head, data);
            printf("Inserted at last successfully!!!\n");

         }
         break;
         case 2:
```

```
            {
                // Print list
                printlist(head);
            }
            break;
            case 3:
            {
                // Create a cycle (for testing)
                int cycle_index;
                printf("Enter the index where you want to create a cycle (1-based): ");
                scanf("%d", &cycle_index);
                createCycle(&head, cycle_index);

            }
            break;
            case 4:
            {
                // Detect cycle
                find_cyclic(&head);

            }
            break;
            case 5:
            {
                // Remove cycle
                remove_cycle(&head);
            }
            break;
            case 6:
            {
                printf("Exiting!!!\n");
            }
            break;
            default:
                printf("Invalid option !! Please try again\n");
        }
    }while(op != 6);
}

void insertLast(Node **head,int data){
    Node *new = (Node *)malloc(sizeof(Node));
    if(new == NULL){
        return;
```

```
    }
    else{
        new->data = data;
        new->link = NULL;

    }

    if(*head == NULL){
        *head = new;
    }
    else{
    Node *temp = *head;
    while(temp->link != NULL){
        temp = temp->link;
    }
    temp->link = new;
    }
}

void printlist(Node *head){

    if (head == NULL) {
        printf("List is empty.\n");
        return;
    }

    while(head != NULL){
        printf("%d->",head->data);
        head = head->link;
    }
    printf("NULL\n");
}

void createCycle(Node **head,int cycle_index){
    if (*head == NULL){
        printf("List is empty, cannot create a cycle.\n");
        return;
    }

    Node *temp = *head;
    Node *cycle_start_node = NULL;
    int index = 1;

    while (temp->link != NULL)
```

```
    {
        if (index == cycle_index)
        {
            cycle_start_node = temp;
        }
        temp = temp->link;
        index++;
    }

    if (cycle_start_node != NULL)
    {
        temp->link = cycle_start_node;
        printf("Cycle created at index %d\n", cycle_index);
    }
    else
    {
        printf("Invalid index. No cycle created.\n");
    }
}

void find_cyclic(Node **head)
{
    if(*head == NULL){
        printf("List is empty\n");
        return;
    }
    Node *fast = *head;
    Node *slow = *head;

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

        if(fast == slow){
            printf("Cycle detected in the list.\n");
            return;
        }
    }
    printf("No cycle found in the list.\n");
}


void remove_cycle(Node **head)
{
```

```
    if (*head == NULL)
    {
        printf("Error: List is empty!\n");
        return;
    }

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

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

        if(slow == fast)
        {

            slow = *head;
            while(slow != fast)
            {
                slow = slow->link;
                fast = fast->link;
            }


            Node *temp = fast;
            while(temp->link != fast)
            {
                temp = temp->link;
            }
            temp->link = NULL;
            printf("Cycle removed successfully.\n");
            return;
        }
    }
    printf("No cycle to remove.\n");
}
```

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

typedef struct node
{
    char name[50];
    int roll_no;
    int class;
    char section;
    int marks[3];
    struct node *link;
}student;

int main()
{
    student *head = NULL;
    printf("Enter details of 5 students:\n");
    for(int i=0; i<5; i++)
    {
        student *new = (student *)malloc(sizeof(student));
        student *temp;
        printf("\nStudent %d\n", i+1);
        printf("Enter the name of student: ");
        scanf(" %[^\n]", new->name);
        printf("Enter the roll no: ");
        scanf(" %d",&new->roll_no);
        printf("Enter the class and section: ");
        scanf("%d %c", &new->class, &new->section);
        printf("Enter the marks of 3 subjects: \n");
        for(int j=0; j<3; j++)
        {
            printf("Subject %d: ", j+1);
            scanf("%d",&new->marks[j]);
        }
        new->link = NULL;

        if(head == NULL)
        {
            head = new;
        }
        else
        {
```

```
            temp=head;

            while(temp->link != NULL)
            {
                temp = temp->link;
            }
            temp->link = new;
        }

    }

    //Display details
    // traversing

    printf("\nStudent details\n");
    student *temp ;
    temp= head;
    int i=1;
    while(temp != NULL)
    {
        printf("\nStudent %d\n", i);
        printf("Name: %s\n", temp->name);
        printf("Roll no: %d\n", temp->roll_no);
        printf("Class %d, Sec: %c\n", temp->class, temp->section);
        printf("Marks: %d %d %d\n", temp->marks[0], temp->marks[1], temp->marks[2]);
        i++;
        temp = temp->link;
    }




    return 0;
}
```