```c
// Name: Ayush Gupta
// Rollno: MNS2025027


// 1. For a given singly linked list, write a program to
remove the duplicate elements from the
// list.
#include <stdio.h>
#include <stdlib.h>

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

void get_element(Node **head)
{
    int val;
    printf("Enter element: ");
    scanf("%d", &val);

    Node *newNode = malloc(sizeof(Node));

    newNode->data = val;
    newNode->next = NULL;

    if (!*head)
    {
        *head = newNode;
        return;
    }
    Node *p = *head;

    while (p->next)
    {
        p = p->next;
    }

    p->next = newNode;
}

void show_list(Node **head)
{
    Node *p = *head;

    if (!*head)
    {
        printf("NULL");
        return;
    }

    while (p)
    {

        printf("%d -> ", p->data);
        p = p->next;
    }
    printf("NULL\n");
}

void remove_duplicate(Node **head)
{
    if (!*head)
        return;

    Node *curr = *head;

    while (curr)
    {
        Node *Next = curr;
        while (Next->next)
        {
            if (Next->next->data == curr->data)
            {
                Node *temp = Next->next;
                Next->next = Next->next->next;
                free(temp);
            }
            else
            {
                Next = Next->next;
            }
        }
        curr = curr->next;
    }
}

int main()
{
    int n;
    Node *head = NULL;
    do
    {
        printf("\n------Menu-------\n");
        printf("1.Enter element\n");
        printf("2.Show List\n");
        printf("3.Remove Duplicate\n");

        printf("8.Exit\n");

        scanf("%d", &n);

        switch (n)
        {
        case 1:
            get_element(&head);
            break;
        case 2:
            show_list(&head);
            break;
        case 3:
            remove_duplicate(&head);
            break;
        }
    } while (n != 8);

    return 0;
}



// 2. Create a singly linked list by adding nodes one by
one in such a way that the resulting
// linked list remains sorted.
#include <stdio.h>
#include <stdlib.h>
```

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

void get_element(Node **head)
{
    int val;
    printf("Enter element: ");
    scanf("%d", &val);

    Node *newNode = malloc(sizeof(Node));

    newNode->data = val;
    newNode->next = NULL;

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

    Node *p = *head;

    while (p->next)
    {
        if (p->next->data > val)
            break;
        if (p->next->data <= val)
            p = p->next;
    }

    newNode->next = p->next;
    p->next = newNode;
}

void show_list(Node **head)
{
    Node *p = *head;

    if (!*head)
    {
        printf("NULL");
        return;
    }

    while (p)
    {

        printf("%d -> ", p->data);
        p = p->next;
    }
    printf("NULL\n");
}

int main()
{
    int n;
    Node *head = NULL;
    do
    {
        printf("\n------Menu-------\n");
        printf("1.Enter element\n");
        printf("2.Show List\n");

        printf("8.Exit\n");

        scanf("%d", &n);

        switch (n)
        {
        case 1:
            get_element(&head);
            break;
        case 2:
            show_list(&head);
            break;
        }
    } while (n != 8);

    return 0;
}



// 3. Given a singly linked list and two integers M and
N, modify the list such that you skip M
// nodes and then delete the next N nodes, repeating this
process until the end of the list.
#include <stdio.h>
#include <stdlib.h>

typedef struct node
```

```c
{
    int data;
    struct node *next;
} Node;

void get_element(Node **head)
{
    int val;
    printf("Enter element: ");
    scanf("%d", &val);

    Node *newNode = malloc(sizeof(Node));

    newNode->data = val;
    newNode->next = NULL;

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

    Node *p = *head;

    while (p->next)
    {
        if (p->next->data > val) break;
        if (p->next->data <= val) p = p->next;
    }

    newNode->next = p->next;
    p->next = newNode;
```

```c
}

void show_list(Node **head)
{
    Node *p = *head;

    if (!*head)
    {
        printf("NULL");
        return;
    }

    while (p)
    {
        printf("%d -> ", p->data);
        p = p->next;
    }
    printf("NULL\n");
}

void skip_and_delete(Node **head)
{
    if (!*head) return;

    int n, m;
    printf("Enter the value of m and n: ");
    scanf("%d%d", &m, &n);

    Node *p = *head;

    while (p)
    {
        int mcount = m, ncount = n;
        while (mcount > 1 && p != NULL)
        {
            p = p->next;
            mcount--;
        }
        if (!p) return;

        Node *curr = p->next;
        while (ncount > 0 && curr != NULL)
        {
            Node *todelete = curr;
            curr = curr->next;
            free(todelete);
            ncount--;
        }
        p->next = curr;
        p = curr;

        if (!p) return;
    }
}
int main()
{
    int n;
    Node *head = NULL;
    do
    {
        printf("\n------Menu-------\n");
        printf("1.Enter element\n");
        printf("2.Show List\n");
        printf("3.Have fun\n");
```

```c
        printf("8.Exit\n");

        scanf("%d", &n);

        switch (n)
        {
            case 1:
                get_element(&head);
                break;
            case 2:
                show_list(&head);
                break;
            case 3:
                skip_and_delete(&head);
                break;
        }
    } while (n != 8);

    return 0;
}
```