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

1. Write a menu driven program for performing stack
operations such a PUSH, POP,
finding top of the stack.*/

#include <stdio.h>

#define max 100

int stack[max];
int top = -1;

void push()
{
    int val;
    if (top >= max - 1)
    {
        printf("Overflow!\n");
        return;
    }
    printf("Value? ");
    scanf("%d", &val);
    stack[++top] = val;
}

void pop()
{
    if (top < 0)
    {
        printf("Underflow!");
        return;
    }
    printf("Popped %d", stack[top--]);
}
void seek()
{
    if (top < 0)
    {
        printf("-1");
        return;
    }
    printf("Top value is %d", stack[top]);
    return;
}

int main()
{
    int n;
    do
    {
        printf("\n---------------");
        printf("\n1.Push ?");
        printf("\n2.Pop ?");
        printf("\n3.Seek ?");
        printf("\n4.Exit\n");
        scanf("%d", &n);

        switch (n)
        {
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            seek();
            break;
        }
    } while (n != 4);

    return 0;
}

// 2. Implement stack operations as in program 1 using
singly linked list
// instead of array.

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

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

Node *top = NULL;

void push()
{
    Node *newNode = malloc(sizeof(Node));

    int val;
    printf("Value?");
    scanf("%d", &val);

    newNode->data = val;
    newNode->next = top;
    top = newNode;
}

void pop()
{
    if (top == NULL)
    {
        printf("-1");
        return;
    }
    Node *temp = top;
    top = top->next;

    printf("Popped %d", temp->data);
    free(temp);
}

void seek()
{
    if (top == NULL)
    {
        printf("-1");
        return;
    }
    printf("Top value %d", top->data);
}
```

```c
int main()
{
    int n;
    do
    {
        printf("\n---------------");
        printf("\n1.Push ?");
        printf("\n2.Pop ?");
        printf("\n3.Seek ?");
        printf("\n4.Exit\n");
        scanf("%d", &n);

        switch (n)
        {
        case 1:
            push();
            break;
        case 2:
            pop();
            break;
        case 3:
            seek();
            break;
        }
    } while (n != 4);

    return 0;
}

/*3. Write a C Program to implement Queue using an array.
The program should be
```

```c
menu-driven, with the option to enqueue, and dequeue, to
check whether the queue is
empty or full, traverse, and exit based on the user's
choice.*/

#include <stdio.h>

#define max 100

int queue[max];
int front = -1, rear = -1;

void enq()
{
    if (rear >= max - 1)
    {
        printf("Overflow!");
        return;
    }

    int val;
    printf("Value?");
    scanf("%d", &val);
    if (front == -1)
        front = 0;
    queue[++rear] = val;
}

void deq()
{
    if (front == -1)
    {
        printf("Empty!");
        return;
    }
    printf("Value at the front is %d", queue[front++]);

    if (front > rear)
    {
        front = -1, rear = -1;
        return;
    }
}

void isempty()
{
    if (front == -1)
    {
        printf("Yup!");
        return;
    }
    printf("NOPE!");
}
void isfull()
{
    if (rear == max - 1)
    {
        printf("Yup!");
        return;
    }
    printf("NOPE!");
}

void traverse()
```

```c
{
    if (front == -1)
    {
        printf("Empty!");
        return;
    }
    for (int i = front; i <= rear; i++)
    {
        printf("[%d] ", queue[i]);
    }
}

int main()
{
    int n;
    do
    {
        printf("\n---------------");
        printf("\n1.Enqueue?");
        printf("\n2.Dequeue?");
        printf("\n3.Full?");
        printf("\n4.Empty?");
        printf("\n5.Traverse?");
        printf("\n6.Exit?\n");
        scanf("%d", &n);

        switch (n)
        {
        case 1:
            enq();
            break;
        case 2:
            deq();
            break;
        case 3:
            isfull();
            break;
        case 4:
            isempty();
            break;
        case 5:
            traverse();
            break;
        }
    } while (n != 6);

    return 0;
}

/*4. Write a C Program to implement Circular Queue using
an array. The program should be
menu-driven, with the option to enqueue, and dequeue, to
check whether the queue is
empty or full, traverse, and exit based on the user's
choice.*/

#include <stdio.h>

#define max 4

int queue[max];
int front = -1, rear = -1;

void enq()
```

```c
{

    if ((rear + 1) % max == front)
    {
        printf("Full!");
        return;
    }
    int val;

    printf("Value? ");
    scanf("%d", &val);

    if (front == -1)
        front = rear = 0;
    else
        rear = (rear + 1) % max;
    queue[rear] = val;
}

void deq()
{
    if (front == -1)
    {
        printf("Empty!");
        return;
    }
    printf("Value at front is %d ", queue[front]);
    if (front == rear)
        front = rear = -1;
    else
        front = (front + 1) % max;
}
```

```c
void traverse()
{
    if (front == -1)
    {
        printf("Empty!");
        return;
    }
    int i = front;
    while (1)
    {
        printf("[%d] ", queue[i]);
        if (i == rear)
            break;
        i = (i + 1) % max;
    }
}

void isfull()
{
    if ((rear + 1) % max == front)
    {
        printf("YUP!!");
        return;
    }
    printf("NOPE!");
}

void isempty()
{
    if (front == -1)
    {
        printf("YUP!");
        return;
    }
    printf("NOPE!");
}

int main()
{
    int n;
    do
    {
        printf("\n---------------");
        printf("\n1.Enqueue?");
        printf("\n2.Dequeue?");
        printf("\n3.Full?");
        printf("\n4.Empty?");
        printf("\n5.Traverse?");
        printf("\n6.Exit?\n");
        scanf("%d", &n);

        switch (n)
        {
        case 1:
            enq();
            break;
        case 2:
            deq();
            break;
        case 3:
            isfull();
            break;
        case 4:
            isempty();
            break;
        case 5:
            traverse();
            break;
        }
    } while (n != 6);

    return 0;
}

/*5. Implement Circular Queue and perform operations on
it using doubly circular linked list
instead of array.*/

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

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

} Node;

Node *front = NULL, *rear = NULL;

void enq()
{
    Node *newNode = malloc(sizeof(Node));
    int val;
    printf("Value? ");
    scanf("%d", &val);

    newNode->data = val;

    if (front == NULL)
    {
        front = rear = newNode;
        newNode->next = newNode->prev = newNode;
        return;
    }

    newNode->next = front;
    newNode->prev = rear;
    rear->next = newNode;
    front->prev = newNode;
    rear = newNode;
}

void deq()
{
    if (front == NULL)
    {
        printf("Empty!");
        return;
    }

    Node *temp = front;

    if (front == rear)
    {
        free(temp);
        front = rear = NULL;
        return;
    }

    front = front->next;
    front->prev = rear;
    rear->next = front;

    printf("Front value is %d", temp->data);
    free(temp);
}

void isempty()
{
    if (front == NULL)
        printf("YUP!");
    else
        printf("NOPE!");
}

void traverse()
{
    if (front == NULL)
        printf("Empty!");
    else
    {
        Node *temp = front;
        do
        {
            printf("[%d] ", temp->data);
            temp = temp->next;
```

```c
        } while ((temp) != front);
    }
}

int main()
{
    int n;
    do
    {
        printf("\n---------------");
        printf("\n1.Enqueue?");
        printf("\n2.Dequeue?");

        printf("\n3.Empty?");
        printf("\n4.Traverse?");
        printf("\n5.Exit?\n");
        scanf("%d", &n);

        switch (n)
        {
        case 1:
            enq();
            break;
        case 2:
            deq();
            break;

        case 3:
            isempty();
            break;
        case 4:
            traverse();
            break;
        }
    } while (n != 5);

    return 0;
}
```