Aim

Implement stack operations (push, pop) using two queues.

Algorithm

- Push: enqueue into queue1.
- Pop: dequeue all except last from queue1 into queue2, then swap queues.

C Code

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

#define SIZE 100

struct Queue {
    int arr[SIZE];
    int front, rear;
};

void init(struct Queue* q) { q->front=q->rear=-1; }
int isEmpty(struct Queue* q) { return q->front==-1; }
void enqueue(struct Queue* q, int val) {
    if (q->rear==SIZE-1) return;
    if (q->front==-1) q->front=0;
    q->arr[++q->rear]=val;
}
int dequeue(struct Queue* q) {
    if (isEmpty(q)) return -1;
    int val=q->arr[q->front];
    if (q->front==q->rear) q->front=q->rear=-1;
    else q->front++;
    return val;
}

struct Stack {
    struct Queue q1, q2;
};

void push(struct Stack* s, int x) { enqueue(&s->q1,x); }
```

```
int pop(struct Stack* s) {
    if (isEmpty(&s->q1)) return -1;
    while (s->q1.front!=s->q1.rear)
        enqueue(&s->q2,dequeue(&s->q1));
    int val=dequeue(&s->q1);
    struct Queue temp=s->q1; s->q1=s->q2; s->q2=temp;
    return val;
}

int main() {
    struct Stack s; init(&s.q1); init(&s.q2);
    push(&s,10); push(&s,20); push(&s,30);
    printf("Popped: %d\n", pop(&s));
    printf("Popped: %d\n", pop(&s));
    return 0;
}
```

Input

Push 10, 20, 30 → Pop twice.

Output

```
Popped: 30
Popped: 20
```

Result

Stack implemented using two queues.