**Assignment**

# a) Linear Queue using arrays:

### 

### *Code:*

#include <stdio.h>

#include <stdbool.h>

#define SIZE 10

struct Queue {

int front, rear;

int items[SIZE];

};

bool isEmpty(struct Queue\* q) {

return q->front == q->rear;

}

bool isFull(struct Queue\* q) {

return (q->rear + 1) % SIZE == q->front;

}

void enqueue(struct Queue\* q, int item) {

if (isFull(q)) {

printf("Queue is full! Cannot enqueue %d\n", item);

return;

}

q->items[q->rear] = item;

q->rear = (q->rear + 1) % SIZE;

}

int dequeue(struct Queue\* q) {

if (isEmpty(q)) {

printf("Queue is empty! Cannot dequeue.\n");

return -1;

}

int item = q->items[q->front];

q->front = (q->front + 1) % SIZE;

return item;

}

bool findElement(struct Queue\* q, int element) {

for (int i = q->front; i != q->rear; i = (i + 1) % SIZE) {

if (q->items[i] == element) {

return true;

}

}

return false;

}

int main() {

struct Queue q;

enqueue(&q, 10);

enqueue(&q, 20);

enqueue(&q, 30);

printf("%d\n", dequeue(&q)); // 10

printf("%d\n", dequeue(&q)); // 20

printf("%d\n", findElement(&q, 30)); // 1 (true)

printf("%d\n", findElement(&q, 40)); // 0 (false)

return 0;

}

### *Output*:



# 1. b) Circular Queue using arrays:

### *Code*:

#include <stdio.h>

#include <stdbool.h>

#define SIZE 10

struct CircularQueue {

int front, rear;

int items[SIZE];

};

bool isEmpty(struct CircularQueue\* q) {

return q->front == q->rear;

}

bool isFull(struct CircularQueue\* q) {

return (q->rear + 1) % SIZE == q->front;

}

void enqueue(struct CircularQueue\* q, int item) {

if (isFull(q)) {

printf("Circular Queue is full! Cannot enqueue %d\n", item);

return;

}

q->items[q->rear] = item;

q->rear = (q->rear + 1) % SIZE;

}

int dequeue(struct CircularQueue\* q) {

if (isEmpty(q)) {

printf("Circular Queue is empty! Cannot dequeue.\n");

return -1;

}

int item = q->items[q->front];

q->front = (q->front + 1) % SIZE;

return item;

}

bool findElement(struct CircularQueue\* q, int element) {

for (int i = q->front; i != q->rear; i = (i + 1) % SIZE) {

if (q->items[i] == element) {

return true;

}

}

return false;

}

int main() {

struct CircularQueue q;

enqueue(&q, 10);

enqueue(&q, 20);

enqueue(&q, 30);

printf("%d\n", dequeue(&q)); // 10

printf("%d\n", dequeue(&q)); // 20

printf("%d\n", findElement(&q, 30)); // 1 (true)

printf("%d\n", findElement(&q, 40)); // 0 (false)

return 0;

}

### *Output*:



# Assignment-2:

### *Code*:

#include <stdio.h>

#include<stdbool.h>

int row, col;

int count = 0;

int a[5];

void place(int i) {

for (int j = 1; j <= 4; j++) {

bool safe = true;

for (int k = 1; k < i; k++) {

if (a[k] == j || (k - i) == (a[k] - j) || (k - i) == (j - a[k])) {

safe = false;

break;

}

}

if (safe) {

a[i] = j;

if (i == 4) {

count++;

printBoard();

} else {

place(i + 1);

}

}

}

}

void printBoard() {

for (int i = 1; i <= 4; i++) {

for (int j = 1; j <= 4; j++) {

if (a[i] == j) {

printf(" Q ");

} else {

printf(" \* ");

}

}

printf("\n");

}

printf("\n");

}

int main() {

row = 4;

col = 4;

place(1);

printf("Number of solutions: %d\n", count);

return 0;

}

### *Output*:

