# TITLE : Event Management System using Stack & Queue

## Abstract

The Event Management System is a console-based C program designed to manage events efficiently using Stack and Queue data structures. Upcoming events are handled using a Queue (FIFO) mechanism, while completed events are stored in a Stack (LIFO) structure. This system helps to organize, process, and review event information in a structured and efficient manner. It demonstrates the practical application of core data structures for real-life event scheduling problems.

## Introduction

Event management involves scheduling, organizing, and tracking multiple activities. Managing events manually can become complex when tracking both upcoming and completed events. This project leverages two key data structures — Queue and Stack — to simulate the process effectively. The Queue handles the scheduling of upcoming events, while the Stack keeps track of completed events. This logical separation ensures easy addition, completion, and retrieval of event data.

## Existing System

In the existing systems, event scheduling and management are usually done manually or using spreadsheets. This leads to inefficiencies, difficulty in tracking completed events, and chances of missing upcoming ones. There is no proper structure to maintain the order in which events are processed.

## Proposed System

The proposed Event Management System automates event handling using Queue and Stack data structures. The Queue ensures events are processed in the order they are added (FIFO), while the Stack keeps track of completed events in reverse order (LIFO). This makes it easier to track both scheduled and finished events efficiently.

## Software Requirements

- Operating System: Windows / Linux  
- Programming Language: C  
- Compiler: GCC / Turbo C / Code::Blocks / Dev C++  
- Editor: Code::Blocks, VS Code, or any C IDE

## Hardware Requirements

- Processor: Intel i3 or higher  
- RAM: Minimum 2 GB  
- Storage: 100 MB free space  
- Input Device: Keyboard  
- Output Device: Monitor

## Full C Code (Short and Simple)

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#define SIZE 50  
  
char queue[SIZE][50];  
int front = -1, rear = -1;  
char stack[SIZE][50];  
int top = -1;  
  
void enqueue(char event[]) {  
 if (rear == SIZE - 1) {  
 printf("Queue Full! Cannot add more events.\n");  
 return;  
 }  
 if (front == -1) front = 0;  
 strcpy(queue[++rear], event);  
 printf("Event added successfully!\n");  
}  
  
void dequeue() {  
 if (front == -1 || front > rear) {  
 printf("No upcoming events to complete.\n");  
 return;  
 }  
 printf("Event Completed: %s\n", queue[front]);  
 strcpy(stack[++top], queue[front]);  
 front++;  
}  
  
void showCompleted() {  
 if (top == -1) {  
 printf("No events completed yet.\n");  
 return;  
 }  
 printf("\n--- Completed Events ---\n");  
 for (int i = top; i >= 0; i--)  
 printf("%s\n", stack[i]);  
}  
  
void showUpcoming() {  
 if (front == -1 || front > rear) {  
 printf("No upcoming events.\n");  
 return;  
 }  
 printf("\n--- Upcoming Events ---\n");  
 for (int i = front; i <= rear; i++)  
 printf("%s\n", queue[i]);  
}  
  
int main() {  
 int choice;  
 char event[50];  
  
 while (1) {  
 printf("\n--- Event Management System ---\n");  
 printf("1. Add Event\n");  
 printf("2. Complete Event\n");  
 printf("3. Show Upcoming Events\n");  
 printf("4. Show Completed Events\n");  
 printf("5. Exit\n");  
 printf("Enter choice: ");  
 scanf("%d", &choice);  
 getchar();  
  
 switch (choice) {  
 case 1:  
 printf("Enter Event Name: ");  
 fgets(event, 50, stdin);  
 event[strcspn(event, "\n")] = 0;  
 enqueue(event);  
 break;  
 case 2:  
 dequeue();  
 break;  
 case 3:  
 showUpcoming();  
 break;  
 case 4:  
 showCompleted();  
 break;  
 case 5:  
 printf("Exiting system. Thank you!\n");  
 exit(0);  
 default:  
 printf("Invalid choice!\n");  
 }  
 }  
 return 0;  
}

## Sample Output

--- Event Management System ---  
1. Add Event  
2. Complete Event  
3. Show Upcoming Events  
4. Show Completed Events  
5. Exit  
Enter choice: 1  
Enter Event Name: Tech Fest  
Event added successfully!  
  
Enter choice: 1  
Enter Event Name: Workshop  
Event added successfully!  
  
Enter choice: 3  
--- Upcoming Events ---  
Tech Fest  
Workshop  
  
Enter choice: 2  
Event Completed: Tech Fest  
  
Enter choice: 4  
--- Completed Events ---  
Tech Fest

## Conclusion

This project successfully demonstrates the use of Stack and Queue data structures in handling real-world event management. The Queue efficiently manages upcoming events in their original order, while the Stack stores completed events for quick retrieval. The system is simple, efficient, and highlights how data structures can solve scheduling and tracking problems effectively.