```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define MAX 100

char stack[MAX];
int top = -1;



void push(char ch) {
   if (top == MAX - 1) {
      printf("Stack overflow!\n");
      return;
   }
   stack[++top] = ch;
}



char pop() {
   if (top == -1) {
      printf("Stack underflow!\n");
      return '\0';
   }
   return stack[top--];
}
```

```
int precedence(char ch) {
   if (ch == '^')
      return 3;
   if (ch == '*' || ch == '/')
      return 2;
   if (ch == '+' || ch == '-')
      return 1;
   return 0;
}

int isOperator(char ch) {
   return ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^';
}


void infixToPostfix(char *infix, char *postfix) {
   int i = 0, j = 0;
   char ch;

   while (infix[i] != '\0') {
      ch = infix[i];

      if (isalnum(ch)) {
         postfix[j++] = ch;
      } else if (ch == '(') {
         push(ch);
      } else if (ch == ')') {
```

```
            while (top != -1 && stack[top] != '(') {
                postfix[j++] = pop();
            }
            pop();
        } else if (isOperator(ch)) {
            while (top != -1 && precedence(stack[top]) >= precedence(ch)) {
                postfix[j++] = pop();
            }
            push(ch);
        }

        i++;
    }


    while (top != -1) {
        postfix[j++] = pop();
    }

    postfix[j] = '\0';
}

int main() {
    char infix[MAX], postfix[MAX];

    printf("Enter infix expression: ");
    scanf("%s", infix);
```

```
    infixToPostfix(infix, postfix);

    printf("Postfix expression: %s\n", postfix);

    return 0;
}
```

2. Reverse a string in stack

```
#include <stdio.h>
#include <string.h>

#define MAX 100

char stack[MAX];
int top = -1;


void push(char ch) {
    if (top == MAX - 1) {
        printf("Stack overflow!\n");
        return;
    }
    stack[++top] = ch;
}


char pop() {
    if (top == -1) {
        printf("Stack underflow!\n");
```

```
        return '\0';
    }
    return stack[top--];
}


void reverseString(char *str) {
    int n = strlen(str);


    for (int i = 0; i < n; i++) {
        push(str[i]);
    }


    for (int i = 0; i < n; i++) {
        str[i] = pop();
    }
}

int main() {
    char str[MAX];

    printf("Enter a string: ");
    scanf("%s", str);

    reverseString(str);
```

```
    printf("Reversed string: %s\n", str);

    return 0;
}
```

3. IMPLEMENTATION OF QUEUE USING ARRAY

```
#include <stdio.h>
#include <stdbool.h>

#define MAX 100

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



bool isEmpty() {
    return front == -1;
}

bool isFull() {
    return rear == MAX - 1;
}

void enqueue(int value) {
    if (isFull()) {
        printf("Queue overflow! Cannot enqueue %d\n", value);
        return;
```

```
    }
    if (front == -1) {
        front = 0;
    }
    queue[++rear] = value;
    printf("%d enqueued successfully.\n", value);
}


int dequeue() {
    if (isEmpty()) {
        printf("Queue underflow! Cannot dequeue.\n");
        return -1;
    }
    int value = queue[front];
    if (front == rear) {
        front = rear = -1;
    } else {
        front++;
    }
    return value;
}

int peek() {
    if (isEmpty()) {
        printf("Queue is empty! No front element.\n");
        return -1;
    }
```

```
    return queue[front];
}

void display() {
    if (isEmpty()) {
        printf("Queue is empty!\n");
        return;
    }
    printf("Queue elements: ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int choice, value;

    while (1) {
        printf("\nQueue Operations:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Peek\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
    switch (choice) {
        case 1:
            printf("Enter value to enqueue: ");
            scanf("%d", &value);
            enqueue(value);
            break;
        case 2:
            value = dequeue();
            if (value != -1) {
                printf("Dequeued: %d\n", value);
            }
            break;
        case 3:
            value = peek();
            if (value != -1) {
                printf("Front element: %d\n", value);
            }
            break;
        case 4:
            display();
            break;
        case 5:
            printf("Exiting...\n");
            return 0;
        default:
            printf("Invalid choice! Please try again.\n");
    }
}
```

```
    return 0;
}
```

4.Simulate a Call Center Queue Create a program to simulate a call center where incoming calls are handled on a first-come, first-served basis. Use a queue to manage call handling and provide options to add, remove, and view calls.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

typedef struct {
    int id;
    char callerName[50];
} Call;

Call queue[MAX];
int front = -1, rear = -1;

int isEmpty() { return front == -1; }
int isFull() { return rear == MAX - 1; }

void addCall(int id, char *name) {
    if (isFull()) return;
    if (front == -1) front = 0;
    rear++;
    queue[rear].id = id;
    strcpy(queue[rear].callerName, name);
}
```

```
void handleCall() {
    if (isEmpty()) return;
    if (front == rear) front = rear = -1;
    else front++;
}

void displayCalls() {
    if (isEmpty()) return;
    for (int i = front; i <= rear; i++) {
        printf("ID: %d, Caller: %s\n", queue[i].id, queue[i].callerName);
    }
}

int main() {
    int choice, id;
    char name[50];
    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d %s", &id, name);
                addCall(id, name);
                break;
            case 2:
                handleCall();
                break;
            case 3:
                displayCalls();
```

```
            break;
        case 4:
            return 0;
    }
  }
}
```

5.Print Job Scheduler Implement a print job scheduler where print requests are queued. Allow users to add new print jobs, cancel a specific job, and print jobs in the order they were added.

```
#include <stdio.h>
#include <string.h>
#define MAX 100

typedef struct {
  int jobId;
  char jobName[50];
} PrintJob;

PrintJob queue[MAX];
int front = -1, rear = -1;

int isEmpty() { return front == -1; }
int isFull() { return rear == MAX - 1; }

void addJob(int id, char *name) {
  if (isFull()) return;
  if (front == -1) front = 0;
```

```
    rear++;
    queue[rear].jobId = id;
    strcpy(queue[rear].jobName, name);
}

void cancelJob(int id) {
    if (isEmpty()) return;
    for (int i = front; i <= rear; i++) {
        if (queue[i].jobId == id) {
            for (int j = i; j < rear; j++) queue[j] = queue[j + 1];
            rear--;
            if (rear < front) front = rear = -1;
            break;
        }
    }
}

void processJobs() {
    if (isEmpty()) return;
    while (!isEmpty()) {
        printf("Printing: %s (ID: %d)\n", queue[front].jobName, queue[front].jobId);
        if (front == rear) front = rear = -1;
        else front++;
    }
}

int main() {
    int choice, id;
```

```
    char name[50];
    while (1) {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d %s", &id, name);
                addJob(id, name);
                break;
            case 2:
                scanf("%d", &id);
                cancelJob(id);
                break;
            case 3:
                processJobs();
                break;
            case 4:
                return 0;
        }
    }
}
```

6.Design a Ticketing System Simulate a ticketing system where people join a queue to buy tickets. Implement functionality for people to join the queue, buy tickets, and display the queue's current state.

```
#include <stdio.h>
#define MAX 100
```

```
int queue[MAX];
int front = -1, rear = -1;

int isEmpty() { return front == -1; }
int isFull() { return rear == MAX - 1; }

void joinQueue(int ticket) {
   if (isFull()) return;
   if (front == -1) front = 0;
   queue[++rear] = ticket;
}

void buyTicket() {
   if (isEmpty()) return;
   if (front == rear) front = rear = -1;
   else front++;
}

void displayQueue() {
   if (isEmpty()) return;
   for (int i = front; i <= rear; i++) printf("%d ", queue[i]);
   printf("\n");
}

int main() {
   int choice, ticket;
   while (1) {
      scanf("%d", &choice);
```

```
        switch (choice) {
            case 1:
                scanf("%d", &ticket);
                joinQueue(ticket);
                break;
            case 2:
                buyTicket();
                break;
            case 3:
                displayQueue();
                break;
            case 4:
                return 0;
        }
    }
}
```