

```

#include <stdio.h>

#include <stdlib.h>

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation error\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void push(struct Node** stack, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *stack;
    *stack = newNode;
    printf("%d pushed to the stack\n", data);
}

int pop(struct Node** stack) {
    if (*stack == NULL) {
        printf("Stack is empty\n");
        exit(EXIT_FAILURE);
    }
    struct Node* temp = *stack;
    *stack = temp->next;
    int poppedValue = temp->data;
    free(temp);
}

```

```

return poppedValue;
}

void display(struct Node* stack) {
if (stack == NULL) {
printf("Stack is empty\n");
return;
}
printf("Stack elements: ");
while (stack != NULL) {
printf("%d ", stack->data);
stack = stack->next;
}
printf("\n");
}

int main() {
struct Node* stack = NULL;

push(&stack, 100);
push(&stack, 200);
push(&stack, 300);

display(stack);

printf("Popped element: %d\n", pop(&stack));

display(stack);

return 0;
}

```

```

100 pushed to the stack
200 pushed to the stack
300 pushed to the stack
Stack elements: 300 200 100
Popped element: 300
Stack elements: 200 100

...Program finished with exit code 0
Press ENTER to exit console.

```

```

#include <stdio.h>

#include <stdlib.h>

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

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

struct Queue {
    struct Node *front, *rear;
};

void initializeQueue(struct Queue* queue) {
    queue->front = queue->rear = NULL;
}

int isEmpty(struct Queue* queue) {
    return (queue->front == NULL);
}

void enqueue(struct Queue* queue, int data) {
    struct Node* newNode = createNode(data);
    if (isEmpty(queue)) {
        queue->front = queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
    }
}

```

```

queue->rear = newNode;
}
printf("%d enqueued to the queue.\n", data);
}

int dequeue(struct Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty. Cannot dequeue.\n");
        exit(EXIT_FAILURE);
    }

    int data = queue->front->data;
    struct Node* temp = queue->front;
    queue->front = queue->front->next;
    free(temp);
    return data;
}

void displayQueue(struct Queue* queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty.\n");
        return;
    }

    struct Node* current = queue->front;
    printf("Queue: ");
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

int main() {
    struct Queue myQueue;
    initializeQueue(&myQueue);

```

```
enqueue(&myQueue, 100);
enqueue(&myQueue, 200);
enqueue(&myQueue, 300);
displayQueue(&myQueue);
printf("Dequeued element: %d\n", dequeue(&myQueue));
displayQueue(&myQueue);
return 0;
}
```

```
100 enqueued to the queue.
200 enqueued to the queue.
300 enqueued to the queue.
Queue: 100 200 300
Dequeued element: 100
Queue: 200 300
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```