```
1. #include <stdio.h>
#include<stdlib.h>
#include<limits.h>
struct Node {
int data;
struct Node* next;
};
struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  if(!newNode) {
    printf("Memory allocation error\n");
    return NULL;
  }
  newNode->data = data;
  newNode->next = NULL;
  return newNode;
}
void push(struct Node** top, int data) {
  struct Node* newNode = createNode(data);
  if (!newNode) {
    return;
  }
  newNode->next = *top;
  *top = newNode;
  printf("%d pushed onto the stack\n", data);
}
int pop(struct Node** top) {
  if (*top == NULL) {
    printf("stack is Empty\n");
    return INT_MIN;
  }
  struct Node* temp = *top;
  int poppedValue = temp->data;
  *top = (*top)->next;
  free(temp);
  return poppedValue;
}
int peek(struct Node* top) {
  if (top == NULL) {
    printf("stack is Empty\n");
    return INT_MIN;
  }
  return top->data;
}
```

```
int beginning(struct Node* top) {
  if (top == NULL) {
    printf("stack is Empty\n");
    return INT_MIN;
  }
  struct Node* current = top;
  while (current->next != NULL) {
    current = current->data;
  }
  return current->data;
}
int main() {
  struct Node* top = NULL;
  push(&top, 10);
  push(&top, 20);
  push(&top, 30);
  push(&top, 40);
  push(&top, 50);
  printf("Top element is %d\n", peek(top));
  printf("Bottom element is %d\n",beginning(top));
  printf("popped element is %d\n", pop(&top));
  printf("popped element is %d\n", pop(&top));
  printf("Top element after popping is %d\n", peek(top));
  return 0;
}
```

```
2.#include<stdio.h>
#define MAX 100
struct Queue {
int items[MAX];
int front, rear;
};
void initialize(struct Queue* q) {
  q->front = -1;
  q->rear = -1;
}
int isFull(struct Queue* q) {
  return q->rear == MAX - 1;
}
int isEmpty(struct Queue* q) {
  return q->front == -1 || q->front > q->rear;
}
void enqueue(struct Queue* q, int value) {
```

```
  if (isFull(q)) {
    printf("Queue is full! cannot insert %d\n", value);
    return;
  }
  if (q->front == -1)
    q->front = 0;
  q->rear++;
  q->items[q->rear] = value;
  printf("Inserted %d\n", value);
}
int dequeue(struct Queue* q) {
  if (isEmpty(q)) {
    printf("Queue is empty! Nothing to remove.\n");
    return -1;
  }
  int value = q->items[q->front];
  q->front++;
  if (q->front > q->rear) {
    q->front = q->rear = -1;
  }
  return value;
}
void display(struct Queue* q) {
  if (isEmpty(q)) {
    printf("Queue is empty!\n");
    return;
  }
  printf("Queue elements: ");
  for (int i = q->front; i <= q->rear; i++) {
    printf("%d ", q->items[i]);
  }
  printf("\n");
}
int main() {
  struct Queue q;
  initialize(&q);

  enqueue(&q, 10);
  enqueue(&q, 20);
  enqueue(&q, 30);

  display(&q);

  printf("Dequeued: %d\n", dequeue(&q));
```

```
  printf("Dequeued: %d\n", dequeue(&q));

  display(&q);

  return 0;
}
```