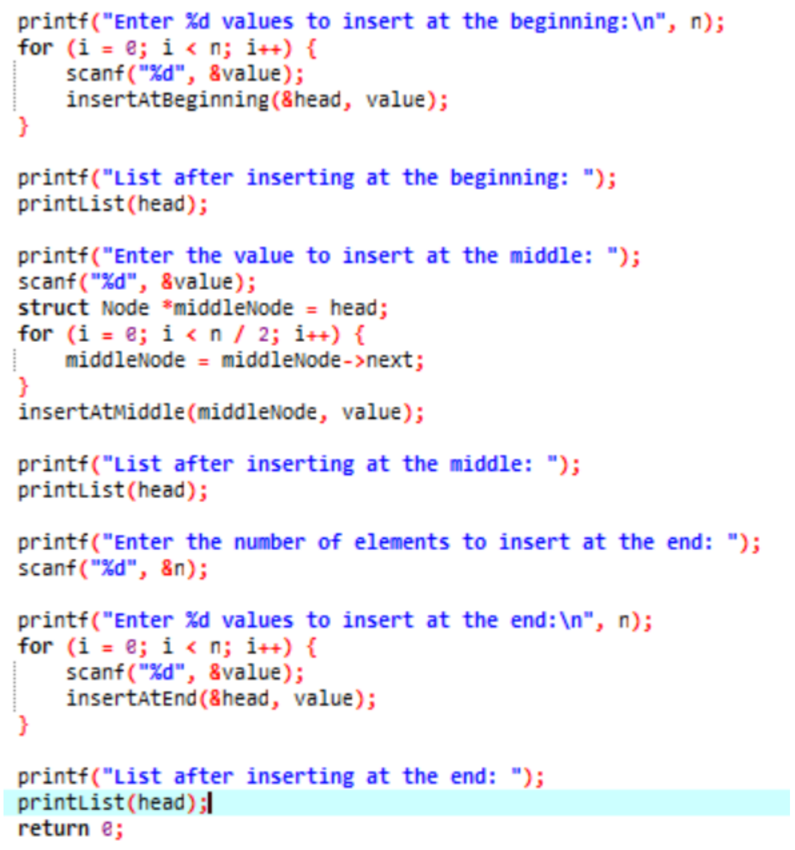
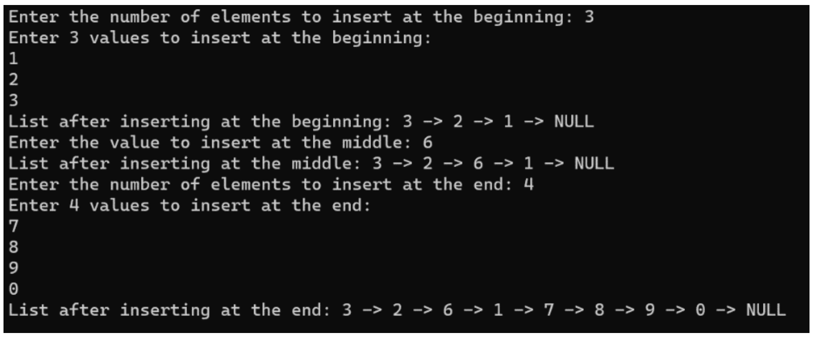
1)write a c program to implement single linked list with the following operation

a) beginning b) middle c) last

```c
#include <stdio.h>
#include <stdlib.h>

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

void insertAtBeginning(struct Node **head, int value) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = *head;
    *head = newNode;
}

void insertAtMiddle(struct Node *prevNode, int value) {
    if (prevNode == NULL) {
        printf("Cannot insert at middle with NULL previous node.\n");
        return;
    }

    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = prevNode->next;
    prevNode->next = newNode;
}

void insertAtEnd(struct Node **head, int value) {
    struct Node *newNode = (struct Node *)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (*head == NULL) {
        *head = newNode;
        return;
    }

    struct Node *current = *head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = newNode;
}

void printList(struct Node *head) {
    struct Node *current = head;
    while (current != NULL) {
        printf("%d -> ", current->data);
        current = current->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node *head = NULL;
    int n, i, value;

    printf("Enter the number of elements to insert at the beginning: ");
    scanf("%d", &n);
```

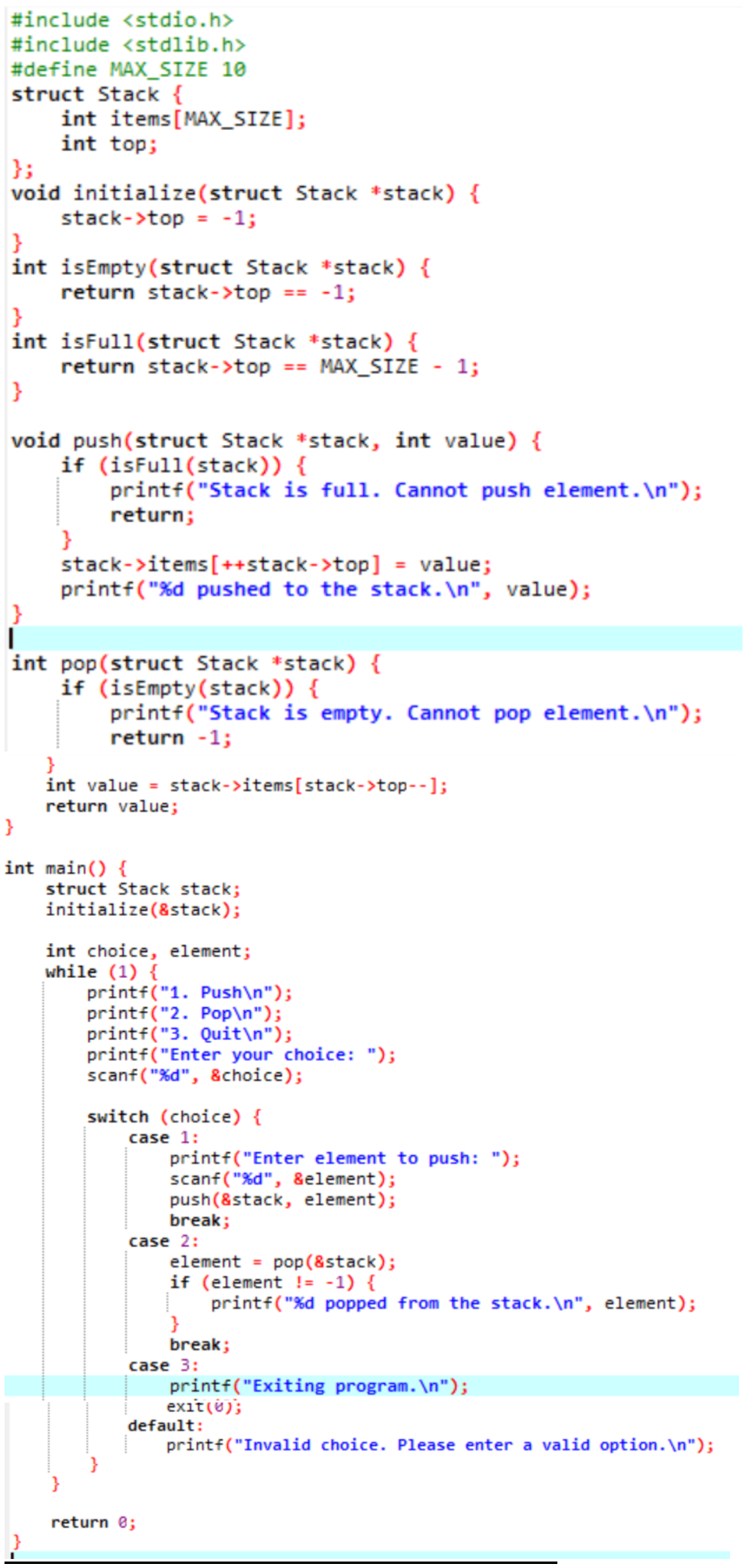


```c
printf("Enter %d values to insert at the beginning:\n", n);
for (i = 0; i < n; i++) {
    scanf("%d", &value);
    insertAtBeginning(&head, value);
}

printf("List after inserting at the beginning: ");
printList(head);

printf("Enter the value to insert at the middle: ");
scanf("%d", &value);
struct Node *middleNode = head;
for (i = 0; i < n / 2; i++) {
    middleNode = middleNode->next;
}
insertAtMiddle(middleNode, value);

printf("List after inserting at the middle: ");
printList(head);

printf("Enter the number of elements to insert at the end: ");
scanf("%d", &n);

printf("Enter %d values to insert at the end:\n", n);
for (i = 0; i < n; i++) {
    scanf("%d", &value);
    insertAtEnd(&head, value);
}

printf("List after inserting at the end: ");
printList(head);
return 0;
```

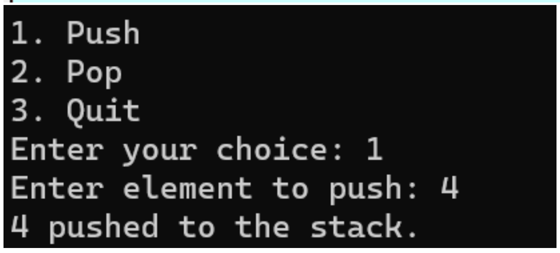


```
Enter the number of elements to insert at the beginning: 3
Enter 3 values to insert at the beginning:
1
2
3
List after inserting at the beginning: 3 -> 2 -> 1 -> NULL
Enter the value to insert at the middle: 6
List after inserting at the middle: 3 -> 2 -> 6 -> 1 -> NULL
Enter the number of elements to insert at the end: 4
Enter 4 values to insert at the end:
7
8
9
0
List after inserting at the end: 3 -> 2 -> 6 -> 1 -> 7 -> 8 -> 9 -> 0 -> NULL
```

2) write a c program to implement STACK data structure PUSH the element into STACK and POP the element into the STACK

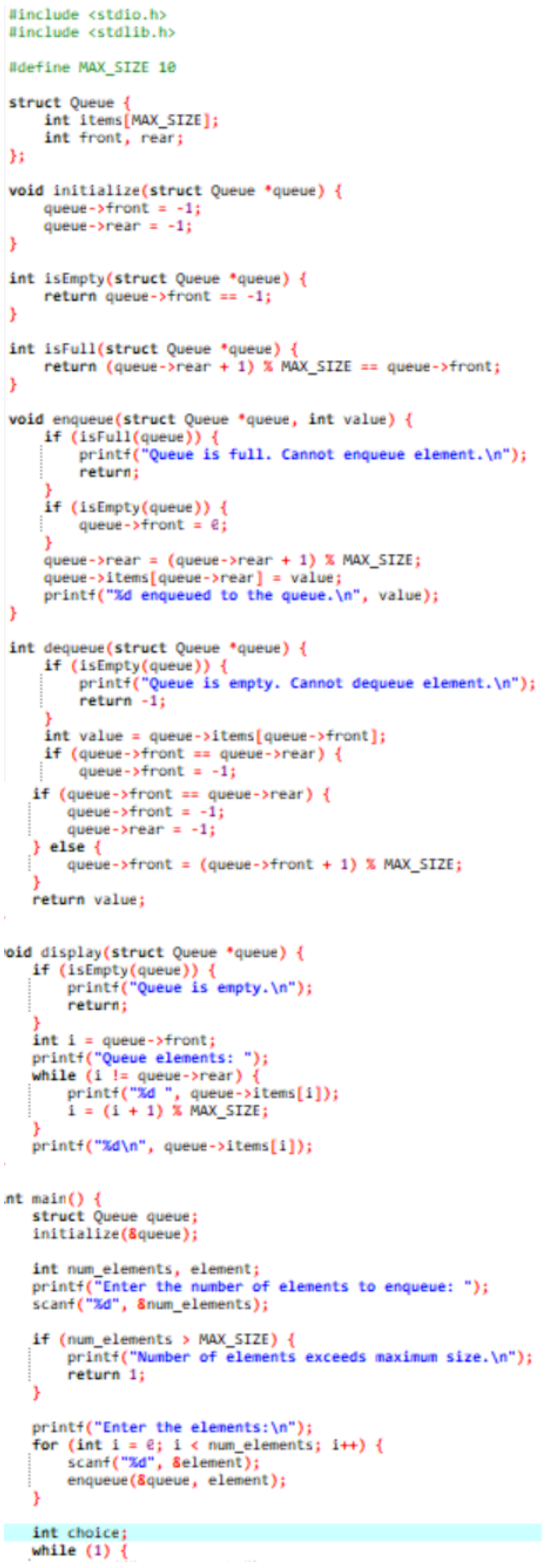


```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 10
struct Stack {
    int items[MAX_SIZE];
    int top;
};
void initialize(struct Stack *stack) {
    stack->top = -1;
}
int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}
int isFull(struct Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}

void push(struct Stack *stack, int value) {
    if (isFull(stack)) {
        printf("Stack is full. Cannot push element.\n");
        return;
    }
    stack->items[++stack->top] = value;
    printf("%d pushed to the stack.\n", value);
}

int pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Cannot pop element.\n");
        return -1;
    }
    int value = stack->items[stack->top--];
    return value;
}

int main() {
    struct Stack stack;
    initialize(&stack);

    int choice, element;
    while (1) {
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter element to push: ");
                scanf("%d", &element);
                push(&stack, element);
                break;
            case 2:
                element = pop(&stack);
                if (element != -1) {
                    printf("%d popped from the stack.\n", element);
                }
                break;
            case 3:
                printf("Exiting program.\n");
                exit(0);
            default:
                printf("Invalid choice. Please enter a valid option.\n");
        }
    }

    return 0;
}
```

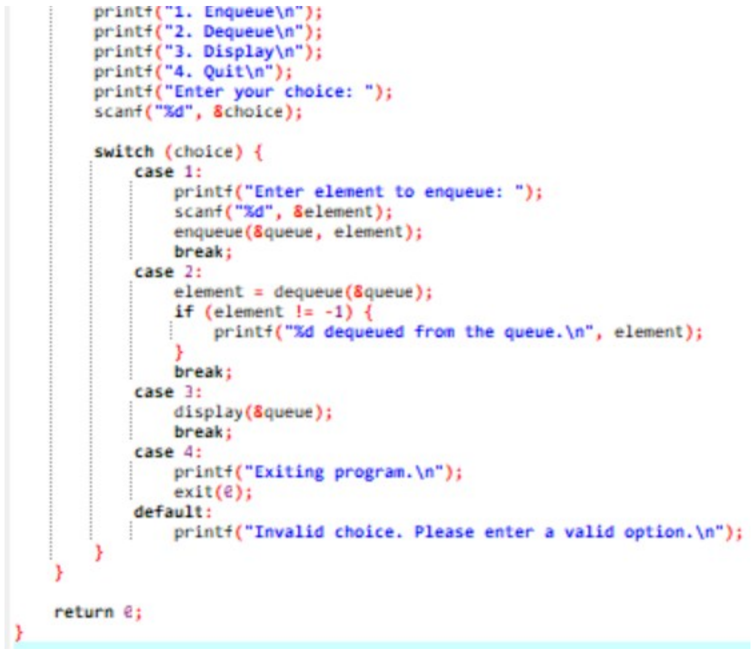


```
1. Push
2. Pop
3. Quit
Enter your choice: 1
Enter element to push: 4
4 pushed to the stack.
```

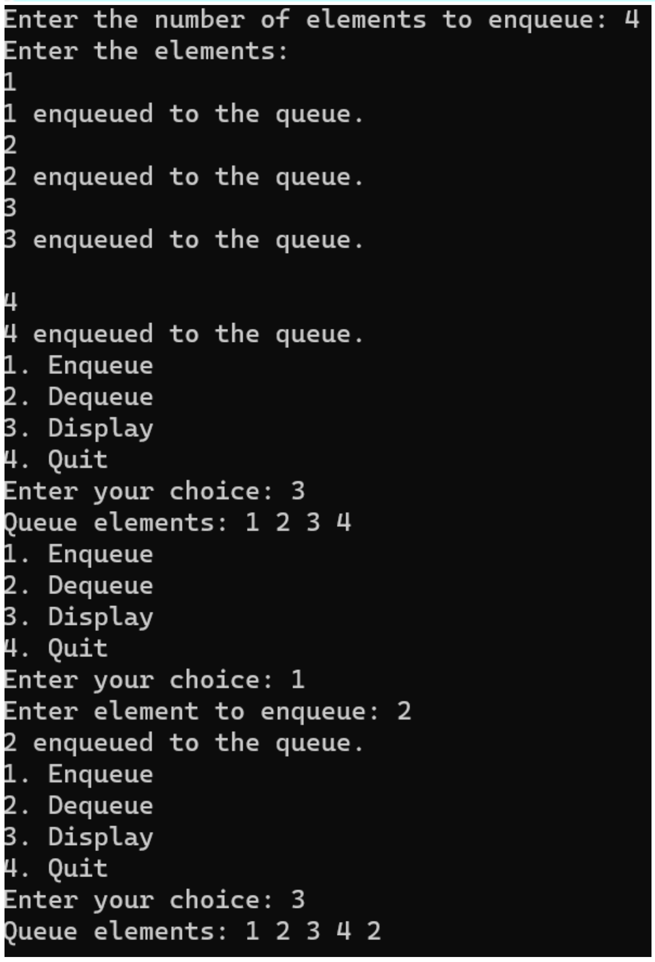
3) write a c program to implement queue data structure with the following enque,deque,display?



```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 10

struct Queue {
    int items[MAX_SIZE];
    int front, rear;
};

void initialize(struct Queue *queue) {
    queue->front = -1;
    queue->rear = -1;
}

int isEmpty(struct Queue *queue) {
    return queue->front == -1;
}

int isFull(struct Queue *queue) {
    return (queue->rear + 1) % MAX_SIZE == queue->front;
}

void enqueue(struct Queue *queue, int value) {
    if (isFull(queue)) {
        printf("Queue is full. Cannot enqueue element.\n");
        return;
    }
    if (isEmpty(queue)) {
        queue->front = 0;
    }
    queue->rear = (queue->rear + 1) % MAX_SIZE;
    queue->items[queue->rear] = value;
    printf("%d enqueued to the queue.\n", value);
}

int dequeue(struct Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty. Cannot dequeue element.\n");
        return -1;
    }
    int value = queue->items[queue->front];
    if (queue->front == queue->rear) {
        queue->front = -1;
    if (queue->front == queue->rear) {
        queue->front = -1;
        queue->rear = -1;
    } else {
        queue->front = (queue->front + 1) % MAX_SIZE;
    }
    return value;

oid display(struct Queue *queue) {
    if (isEmpty(queue)) {
        printf("Queue is empty.\n");
        return;
    }
    int i = queue->front;
    printf("Queue elements: ");
    while (i != queue->rear) {
        printf("%d ", queue->items[i]);
        i = (i + 1) % MAX_SIZE;
    }
    printf("%d\n", queue->items[i]);

nt main() {
    struct Queue queue;
    initialize(&queue);

    int num_elements, element;
    printf("Enter the number of elements to enqueue: ");
    scanf("%d", &num_elements);

    if (num_elements > MAX_SIZE) {
        printf("Number of elements exceeds maximum size.\n");
        return 1;
    }

    printf("Enter the elements:\n");
    for (int i = 0; i < num_elements; i++) {
        scanf("%d", &element);
        enqueue(&queue, element);
    }

    int choice;
    while (1) {
```

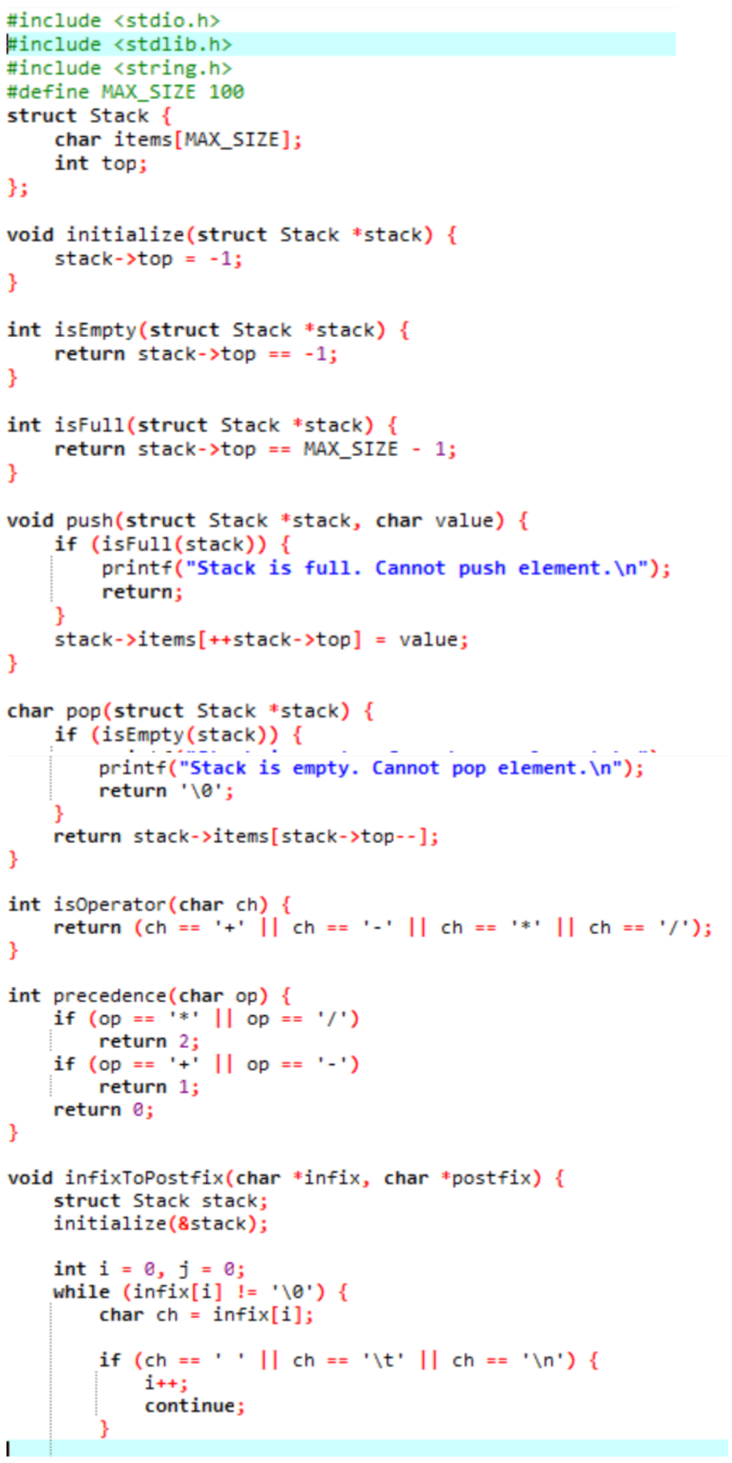


```c
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter element to enqueue: ");
                scanf("%d", &element);
                enqueue(&queue, element);
                break;
            case 2:
                element = dequeue(&queue);
                if (element != -1) {
                    printf("%d dequeued from the queue.\n", element);
                }
                break;
            case 3:
                display(&queue);
                break;
            case 4:
                printf("Exiting program.\n");
                exit(0);
            default:
                printf("Invalid choice. Please enter a valid option.\n");
        }
    }

    return 0;
}
```

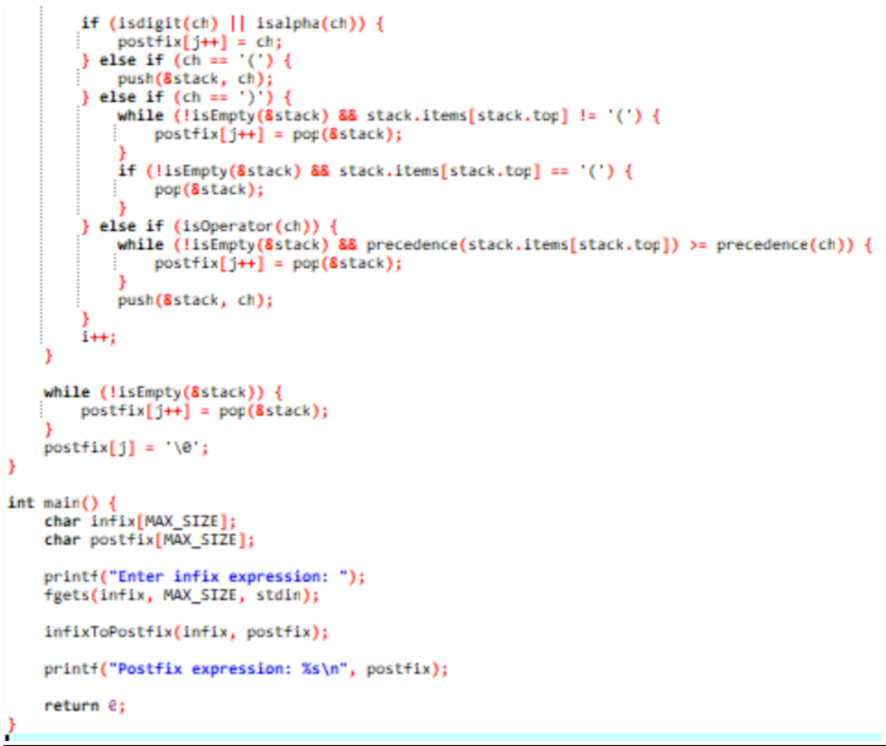


```
Enter the number of elements to enqueue: 4
Enter the elements:
1
1 enqueued to the queue.
2
2 enqueued to the queue.
3
3 enqueued to the queue.

4
4 enqueued to the queue.
1. Enqueue
2. Dequeue
3. Display
4. Quit
Enter your choice: 3
Queue elements: 1 2 3 4
1. Enqueue
2. Dequeue
3. Display
4. Quit
Enter your choice: 1
Enter element to enqueue: 2
2 enqueued to the queue.
1. Enqueue
2. Dequeue
3. Display
4. Quit
Enter your choice: 3
Queue elements: 1 2 3 4 2
```

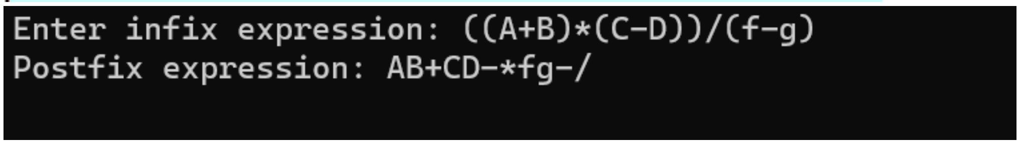
4) write a c program to convert infix expression to postfix expression using STACK?



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_SIZE 100
struct Stack {
    char items[MAX_SIZE];
    int top;
};

void initialize(struct Stack *stack) {
    stack->top = -1;
}

int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}

int isFull(struct Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}

void push(struct Stack *stack, char value) {
    if (isFull(stack)) {
        printf("Stack is full. Cannot push element.\n");
        return;
    }
    stack->items[++stack->top] = value;
}

char pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Cannot pop element.\n");
        return '\0';
    }
    return stack->items[stack->top--];
}

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

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

void infixToPostfix(char *infix, char *postfix) {
    struct Stack stack;
    initialize(&stack);

    int i = 0, j = 0;
    while (infix[i] != '\0') {
        char ch = infix[i];

        if (ch == ' ' || ch == '\t' || ch == '\n') {
            i++;
            continue;
        }
```

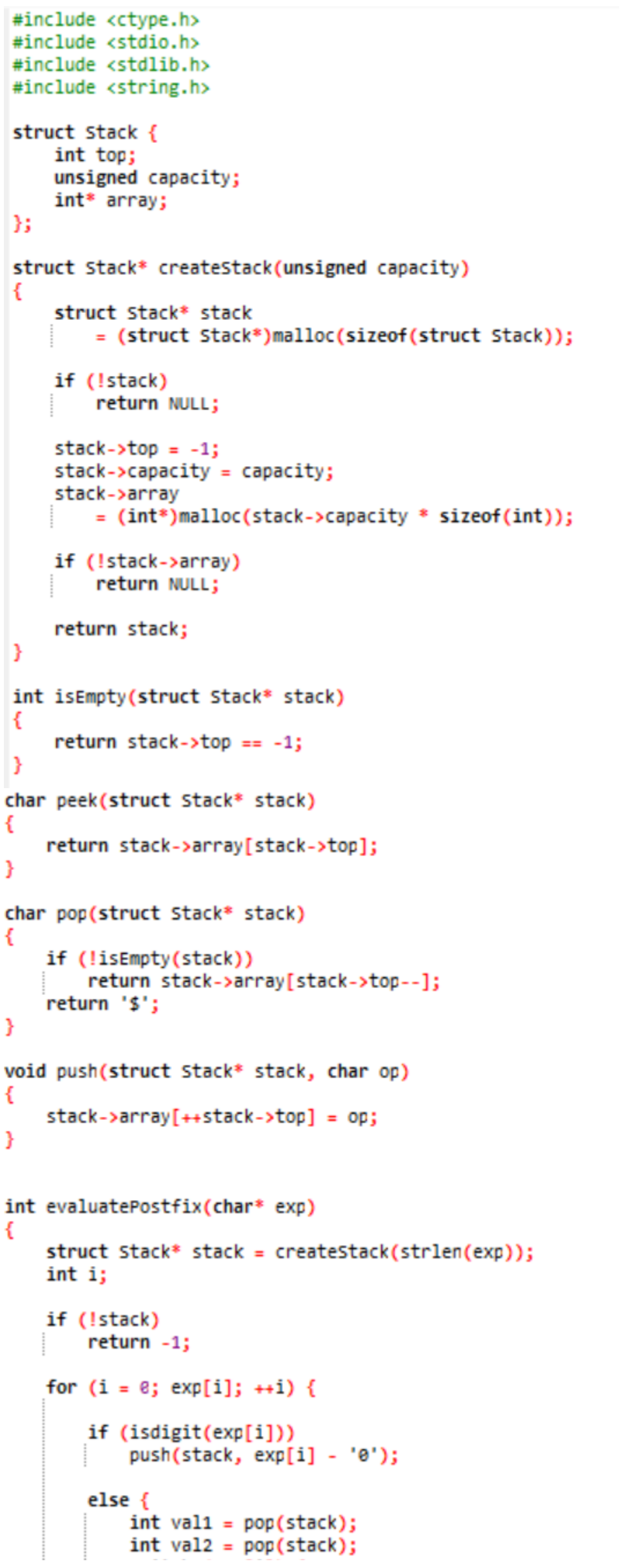


```c
        if (isdigit(ch) || isalpha(ch)) {
            postfix[j++] = ch;
        } else if (ch == '(') {
            push(&stack, ch);
        } else if (ch == ')') {
            while (!isEmpty(&stack) && stack.items[stack.top] != '(') {
                postfix[j++] = pop(&stack);
            }
            if (!isEmpty(&stack) && stack.items[stack.top] == '(') {
                pop(&stack);
            }
        } else if (isOperator(ch)) {
            while (!isEmpty(&stack) && precedence(stack.items[stack.top]) >= precedence(ch)) {
                postfix[j++] = pop(&stack);
            }
            push(&stack, ch);
        }
        i++;
    }

    while (!isEmpty(&stack)) {
        postfix[j++] = pop(&stack);
    }
    postfix[j] = '\0';
}

int main() {
    char infix[MAX_SIZE];
    char postfix[MAX_SIZE];

    printf("Enter infix expression: ");
    fgets(infix, MAX_SIZE, stdin);

    infixToPostfix(infix, postfix);

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

    return 0;
}
```

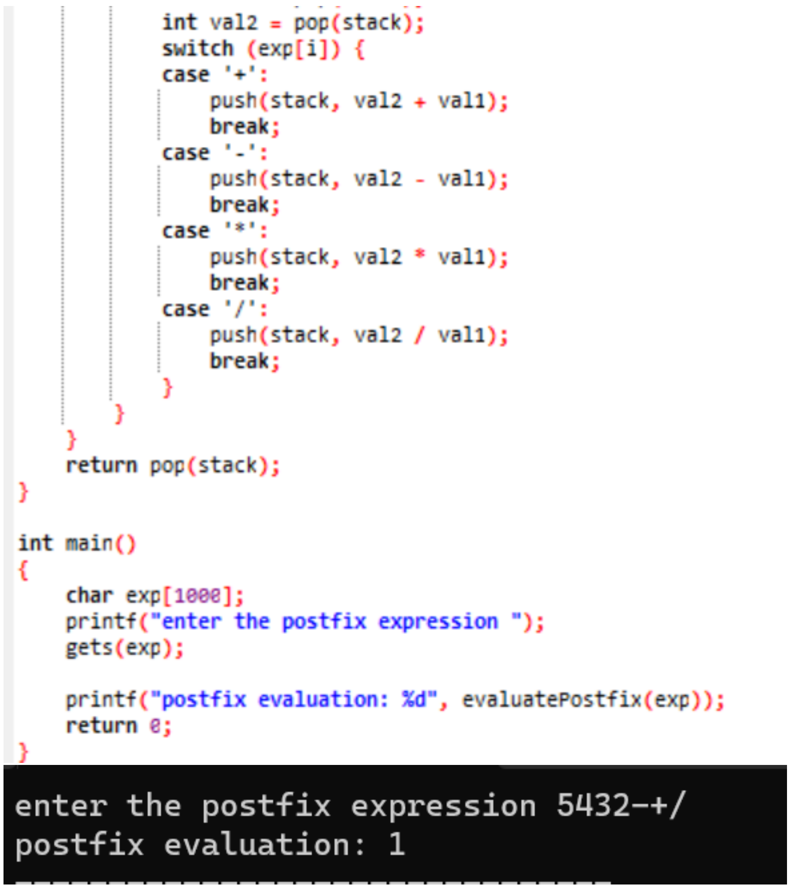


```
Enter infix expression: ((A+B)*(C-D))/(f-g)
Postfix expression: AB+CD-*fg-/
```

5) write a c program to evaluate the given postfix expression using STACK?



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

struct Stack {
    int top;
    unsigned capacity;
    int* array;
};

struct Stack* createStack(unsigned capacity)
{
    struct Stack* stack
        = (struct Stack*)malloc(sizeof(struct Stack));

    if (!stack)
        return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array
        = (int*)malloc(stack->capacity * sizeof(int));

    if (!stack->array)
        return NULL;

    return stack;
}

int isEmpty(struct Stack* stack)
{
    return stack->top == -1;
}
char peek(struct Stack* stack)
{
    return stack->array[stack->top];
}

char pop(struct Stack* stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}

void push(struct Stack* stack, char op)
{
    stack->array[++stack->top] = op;
}


int evaluatePostfix(char* exp)
{
    struct Stack* stack = createStack(strlen(exp));
    int i;

    if (!stack)
        return -1;

    for (i = 0; exp[i]; ++i) {

        if (isdigit(exp[i]))
            push(stack, exp[i] - '0');

        else {
            int val1 = pop(stack);
            int val2 = pop(stack);
```



```
            int val2 = pop(stack);
            switch (exp[i]) {
            case '+':
                push(stack, val2 + val1);
                break;
            case '-':
                push(stack, val2 - val1);
                break;
            case '*':
                push(stack, val2 * val1);
                break;
            case '/':
                push(stack, val2 / val1);
                break;
            }
        }
    }
    return pop(stack);
}

int main()
{
    char exp[1000];
    printf("enter the postfix expression ");
    gets(exp);

    printf("postfix evaluation: %d", evaluatePostfix(exp));
    return 0;
}
```

```
enter the postfix expression 5432-+/
postfix evaluation: 1
```