```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>   // for isdigit()

#define MAX 100      // maximum size of stack

/* ------------ STACK IMPLEMENTATION ------------ */

typedef struct {
    int arr[MAX];
    int top;
} Stack;

// Initialize stack
void initStack(Stack *s) {
    s->top = -1;
}

// Check if stack is empty
int isEmpty(Stack *s) {
    return (s->top == -1);
}

// Check if stack is full
int isFull(Stack *s) {
    return (s->top == MAX - 1);
}

// Push an element onto stack
void push(Stack *s, int value) {
    if (isFull(s)) {
        printf("Stack Overflow! Cannot push %d\n", value);
        return;
    }
    s->arr[++(s->top)] = value;
    printf("%d pushed onto stack.\n", value);
}

// Pop an element from stack
int pop(Stack *s) {
    if (isEmpty(s)) {
        printf("Stack Underflow! Nothing to pop.\n");
        return -1;   // sentinel value
    }
    return s->arr[(s->top)--];
}

// Peek top element of stack
```

```
int peek(Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty. No top element.\n");
        return -1;
    }
    return s->arr[s->top];
}

// Display stack elements
void display(Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty.\n");
        return;
    }
    printf("Stack elements (top to bottom): ");
    for (int i = s->top; i >= 0; i--) {
        printf("%d ", s->arr[i]);
    }
    printf("\n");
}

/* ------------ APPLICATION: POSTFIX EVALUATION ------------ */
/*
   Example postfix expression:
   "23*54*+9-"
   Meaning:
   2 3 * 5 4 * + 9 -
*/

int applyOperator(int a, int b, char op) {
    switch (op) {
        case '+': return a + b;
        case '-': return a - b;
        case '*': return a * b;
        case '/':
            if (b == 0) {
                printf("Error: Division by zero!\n");
                exit(1);
            }
            return a / b;
        default:
            printf("Error: Unknown operator %c\n", op);
            exit(1);
    }
}

// Evaluate postfix expression using stack
int evaluatePostfix(const char *expr) {
```

```
    Stack s;
    initStack(&s);

    int i = 0;
    char ch;

    while ((ch = expr[i]) != '\0' && ch != '\n') {
        if (ch == ' ') {
            // ignore spaces
            i++;
            continue;
        }

        if (isdigit((unsigned char)ch)) {
            // if operand is a single-digit number
            int value = ch - '0';
            push(&s, value);
        } else {
            // operator: pop two operands
            int b = pop(&s);
            int a = pop(&s);
            if (a == -1 || b == -1) {
                printf("Error: Invalid postfix expression.\n");
                return -1;
            }
            int result = applyOperator(a, b, ch);
            push(&s, result);
        }
        i++;
    }

    int finalResult = pop(&s);
    if (!isEmpty(&s)) {
        printf("Warning: Extra operands in expression.\n");
    }
    return finalResult;
}

/* ------------ MAIN MENU ------------ */

int main() {
    Stack s;
    initStack(&s);

    int choice, value;
    char expr[200];

    while (1) {
```

```
        printf("\n===== STACK MENU =====\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek (Top Element)\n");
        printf("4. Display Stack\n");
        printf("5. Evaluate Postfix Expression (Application)\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to push: ");
                scanf("%d", &value);
                push(&s, value);
                break;

            case 2: {
                int popped = pop(&s);
                if (popped != -1)
                    printf("Popped element: %d\n", popped);
                break;
            }

            case 3: {
                int topVal = peek(&s);
                if (topVal != -1)
                    printf("Top element: %d\n", topVal);
                break;
            }

            case 4:
                display(&s);
                break;

            case 5:
                // flush leftover newline from previous scanf
                getchar();
                printf("Enter postfix expression (single-digit operands, no spaces or with
spaces):\n");
                fgets(expr, sizeof(expr), stdin);
                {
                    int result = evaluatePostfix(expr);
                    if (result != -1)
                        printf("Result of postfix expression: %d\n", result);
                }
                break;
```

```
            case 6:
                printf("Exiting...\n");
                exit(0);

            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```