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

#define MAX 100 // Define the maximum size of the stack

// Define the stack structure
typedef struct Stack {
    int top;
    int arr[MAX];
} Stack;

// Function to initialize the stack
void initializeStack(Stack *stack) {
    stack->top = -1; // Stack is initially empty
}

// Function to check if the stack is empty
int isEmpty(Stack *stack) {
    return stack->top == -1;
}

// Function to check if the stack is full
int isFull(Stack *stack) {
    return stack->top == MAX - 1;
}

// Function to push an element onto the stack
void push(Stack *stack, int value) {
    if (isFull(stack)) {
        printf("Stack overflow. Unable to push %d\n", value);
        return;
    }
    stack->arr[++stack->top] = value;
}

// Function to pop an element from the stack
int pop(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow. Unable to pop\n");
        return -1; // Returning -1 to indicate underflow
    }
    return stack->arr[stack->top--];
}

// Function to get the top element without removing it
int peek(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Unable to peek\n");
```

```
        return -1;
    }
    return stack->arr[stack->top];
}

// Function to print all elements in the stack
void printStack(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty\n");
        return;
    }
    printf("Stack elements are: ");
    for (int i = 0; i <= stack->top; i++) {
        printf("%d ", stack->arr[i]);
    }
    printf("\n");
}

// Main function to demonstrate stack operations
int main() {
    Stack stack;
    initializeStack(&stack);

    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);

    printStack(&stack);

    printf("Top element is: %d\n", peek(&stack));

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

    printStack(&stack);

    return 0;
}
```