

# Stack implementation

## Array and linked list implementation:

### Array implementation:

#### Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define MAX 100

typedef struct {

    int top;

    int arr[MAX];

} Stack;

void initStack(Stack* stack) {

    stack->top = -1;

}

int isFull(Stack* stack) {

    return stack->top == MAX - 1;

}

int isEmpty(Stack* stack) {

    return stack->top == -1;

}

void push(Stack* stack, int value) {

    if (isFull(stack)) {

        printf("Stack overflow! Cannot push %d\n", value);

        return;

    }

    stack->arr[++stack->top] = value;

    printf("%d pushed to stack\n", value);

}

int pop(Stack* stack) {

    if (isEmpty(stack)) {

        printf("Stack underflow! Cannot pop\n");

        return INT_MIN;

    }

}
```

```

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

int peek(Stack* stack) {

    if (isEmpty(stack)) {

        printf("Stack is empty! Nothing to peek\n");

        return INT_MIN;

    }

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

int main() {

    Stack stack;

    initStack(&stack);

    push(&stack, 10);

    push(&stack, 20);

    push(&stack, 30);

    printf("%d popped from stack\n", pop(&stack));

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

    return 0;
}

```

## Output:

```

10 pushed to stack

20 pushed to stack

30 pushed to stack

30 popped from stack

Top element is 20

```

## Linked list implementation:

## Code:

```

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

    int data;

    struct Node* next;

} Node;

Node* createNode(int data) {

```

```
Node* newNode = (Node*)malloc(sizeof(Node));

if (!newNode) {

    printf("Memory allocation error\n");

    exit(1);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

void push(Node** top, int data) {

    Node* newNode = createNode(data);

    newNode->next = *top;

    *top = newNode;

    printf("%d pushed to stack\n", data);

}

int pop(Node** top) {

    if (*top == NULL) {

        printf("Stack underflow! Cannot pop\n");

        return -1;

    }

    Node* temp = *top;

    int poppedValue = temp->data;

    *top = (*top)->next;

    free(temp);

    return poppedValue;

}

int peek(Node* top) {

    if (top == NULL) {

        printf("Stack is empty! Nothing to peek\n");

        return -1;

    }

    return top->data;

}

int main() {

    Node* stack = NULL;
```

```
    push(&stack, 10);

    push(&stack, 20);

    push(&stack, 30);

    printf("%d popped from stack\n", pop(&stack));

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

    return 0;

}
```

## Output:

10 pushed to stack

20 pushed to stack

30 pushed to stack

30 popped from stack

Top element is 20