Stack using Array

```
#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

struct Stack {

        int top;

        unsigned capacity;

        int* array;

};

struct Stack* createStack(unsigned capacity)

{

        struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));

        stack->capacity = capacity;

        stack->top = -1;

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

        return stack;

}

int isFull(struct Stack* stack)

{

        return stack->top == stack->capacity - 1;

}

int isEmpty(struct Stack* stack)

{

        return stack->top == -1;
```

```
}

void push(struct Stack* stack, int item)

{

        if (isFull(stack))

                return;

        stack->array[++stack->top] = item;

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

}

int pop(struct Stack* stack)

{

        if (isEmpty(stack))

                return INT_MIN;

        return stack->array[stack->top--];

}

int peek(struct Stack* stack)

{

        if (isEmpty(stack))

                return INT_MIN;

        return stack->array[stack->top];

}

int main()

{

        struct Stack* stack = createStack(100);

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

        push(stack, 30);

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

        return 0;

}
```

Output:

```
10 pushed to stack

20 pushed to stack

30 pushed to stack

30 popped from stack
```

Stack using Linked List:

```
#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 == NULL)

                return NULL;

        newNode->data = data;
```

```
    newNode->next = NULL;

    return newNode;

}

int insertBeforeHead(node** head, int data)

{

    node* newNode = createNode(data);

        if (!newNode)

        return -1;

    if (*head == NULL) {

        *head = newNode;

        return 0;

    }


    newNode->next = *head;

    *head = newNode;

    return 0;

}

int deleteHead(node** head)

{

    node* temp = *head;

    *head = (*head)->next;

    free(temp);

    return 0;

}
```

```
int isEmpty(node** stack) { return *stack == NULL; }

void push(node** stack, int data)

{

        if (insertBeforeHead(stack, data)) {

                printf("Stack Overflow!\n");

        }

}

int pop(node** stack)

{

                if (isEmpty(stack)) {

                printf("Stack Underflow\n");

                return -1;

        }


        // deleting the head.

        deleteHead(stack);

}

int peek(node** stack)

{

        if (!isEmpty(stack))

                return (*stack)->data;

        else

                return -1;

}
```

```
void printStack(node** stack)

{

        node* temp = *stack;

        while (temp != NULL) {

                printf("%d-> ", temp->data);

                temp = temp->next;

        }

        printf("\n");

}

int main()

{

        node* stack = NULL;

        push(&stack, 10);

        push(&stack, 20);

        push(&stack, 30);

        push(&stack, 40);

        push(&stack, 50);

        printf("Stack: ");

        printStack(&stack);

        pop(&stack);

        pop(&stack);

          printf("\nStack: ");

        printStack(&stack);

        return 0;
}
```

Output:

```
Stack: 50-> 40-> 30-> 20-> 10->

Stack: 30-> 20-> 10->
```