**DATASTRUCTURE**

**PROGRAMS:**

**1.Stack using Array Implementation.**

*#include <stdio.h>*

*#include <stdlib.h>*

*#define MAX 100*

*typedef struct {*

*int* *items[MAX];*

*int top;*

*} Stack;*

*void* *initialize(Stack \*s) {*

*s->top = -1;*

*}*

*int isFull(Stack \*s) {*

*return s->top == MAX - 1;*

*}*

*int isEmpty(Stack \*s) {*

*return s->top == -1;*

*}*

*void* *push(Stack \*s, int value) {*

*if (isFull(s)) {*

*printf("Stack overflow\n");*

*} else {*

*s->items[++(s->top)] = value;*

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

*}*

*}*

*int* *pop(Stack \*s) {*

*if (isEmpty(s)) {*

*printf("Stack underflow\n");*

*return -**1; // Returning -1 to indicate stack is empty*

*} else {*

*return s->items[(s->**top)--];*

*}*

*}*

*int* *peek(Stack \*s) {*

*if (isEmpty(s)) {*

*printf("Stack is empty\n");*

*return -**1; // Returning -1 to indicate stack is empty*

*} else {*

*return s->items[s->top];*

*}*

*}*

*int* *main() {*

*Stack s;*

*initialize(&s);*

*push(&s, 10);*

*push(&s, 20);*

*push(&s, 30);*

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

*printf("Elements: \n");*

*while (!isEmpty(&s)) {*

*printf("%d\n", pop(&s));*

*}*

*printf("Stack empty: %s\n", isEmpty(&s**) ? "true**" : "false");*

*return 0;*

*}*

**OUTPUT:**

*10 pushed to stack*

*20 pushed to stack*

*30 pushed to stack*

*Top element is 30*

*Elements:*

*30*

*20*

*10*

*Stack empty: true*

**2.Stack using Linked List Implementation.**

*#include <stdio.h>*

*#include <stdlib.h>*

*typedef struct Node {*

*int data;*

*struct Node\* next;*

*} Node;*

*Node\* newNode(int data) {*

*Node\* stackNode = (Node**\*)malloc(sizeof(Node));*

*stackNode->data = data;*

*stackNode->next = NULL;*

*return stackNode;*

*}*

*int isEmpty(Node\* root) {*

*return !root;*

*}*

*void* *push(Node\*\* root, int data) {*

*Node\* stackNode = newNode(data);*

*stackNode->next = \*root;*

*\*root = stackNode;*

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

*}*

*int* *pop(Node\*\* root) {*

*if (isEmpty(\*root)) {*

*printf("Stack underflow\n");*

*return -1;*

*}*

*Node\* temp = \*root;*

*\*root = (\*root)->next;*

*int popped = temp->data;*

*free(temp);*

*return popped;*

*}*

*int* *peek(Node\* root) {*

*if (isEmpty(root)) {*

*printf("Stack is empty\n");*

*return -1;*

*}*

*return root->data;*

*}*

*int* *main() {*

*Node\* root = NULL;*

*push(&root, 10);*

*push(&root, 20);*

*push(&root, 30);*

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

*printf("Elements: \n");*

*while (!isEmpty(root)) {*

*printf("%d\n", pop(&root));*

*}*

*printf("Stack empty: %s\n", isEmpty(root**) ? "true**" : "false");*

*return 0;*

*}*

**OUTPUT:**

*10 pushed to stack*

*20 pushed to stack*

*30 pushed to stack*

*Top element is 30*

*Elements:*

*30*

*20*

*10*

*Stack empty: true*