DATA STRUCTURE

CSA - 0390, 26/07/2024, DAY - 2

1) WRITE A C PROGRAMMING FOR STACK USING ARRAY.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
struct Stack {
    int arr[MAX];
    int top;
};
void initialize(struct Stack* stack) {
    stack->top = -1;
}
int isEmpty(struct Stack* stack) {
    return stack->top == -1;
}
int isFull(struct Stack* stack) {
    return stack->top == MAX - 1;
}
void push(struct Stack* stack, int value) {
    if (isFull(stack)) {
        printf("Stack overflow! Cannot push %d¥n", value);
        return;
    }
    stack->arr[++stack->top] = value;
    printf("%d pushed onto stack¥n", value);
}
int pop(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow! Cannot pop¥n");
        return -1;
    }
    return stack->arr[stack->top--];
}
int peek(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty!¥n");
        return -1;
    }
    return stack->arr[stack->top];
}
void display(struct Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty!¥n");
        return;
    }
    printf("Stack elements: ");
    for (int i = 0; i <= stack->top; i++) {
        printf("%d ", stack->arr[i]);
    }
    printf("¥n");
}
int main() {
    struct Stack stack;
    initialize(&stack);
    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);
    display(&stack);
    printf("Top element is %d¥n", peek(&stack));
    printf("Popped element is %d¥n", pop(&stack));
    display(&stack);
    return 0;
}
```

OUTPUT :

```
10 pushed onto stack
20 pushed onto stack
30 pushed onto stack
Stack elements: 10 20 30
Top element is 30
Popped element is 30
Stack elements: 10 20
```

2) WRITE A C PROGRAMMING FOR STACK FOR USING LINKED LIST.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* newNode(int data) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;
    node->next = NULL;
    return node;
}
int isEmpty(struct Node* root) {
    return !root;
}
void push(struct Node** root, int data) {
    struct Node* node = newNode(data);
    node->next = *root;
    *root = node;
    printf("%d pushed onto stack¥n", data);
}
int pop(struct Node** root) {
    if (isEmpty(*root)) {
        printf("Stack underflow! Cannot pop¥n");
        return -1;
    }
    struct Node* temp = *root;
    *root = (*root)->next;
    int popped = temp->data;
    free(temp);
    return popped;
}
int peek(struct Node* root) {
    if (isEmpty(root)) {
        printf("Stack is empty!¥n");
        return -1;
    }
    return root->data;
}
void display(struct Node* root) {
    if (isEmpty(root)) {
        printf("Stack is empty!¥n");
        return;
    }
    struct Node* temp = root;
    printf("Stack elements: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("¥n");
}
int main() {
    struct Node* root = NULL;
    push(&root, 10);
    push(&root, 20);
    push(&root, 30);
    display(root);
    printf("Top element is %d¥n", peek(root));
    printf("Popped element is %d¥n", pop(&root));
    display(root);
    return 0;
}
```

OUTPUT :

```
10 pushed onto stack
20 pushed onto stack
30 pushed onto stack
Stack elements: 30 20 10
Top element is 30
Popped element is 30
Stack elements: 20 10
```