DATA STRUCTURE

DAY 03 , 26/07/24 , CSA0390

1 . write a c programming of stack of using array .

```
#include <stdio.h>

#include <stdlib.h>

#define MAX_SIZE 100

int stack[MAX_SIZE];

int top = -1;

void push(int value) {

        if (top == MAX_SIZE - 1) {

                printf("Stack Overflow\n");

                return;

        }

        stack[++top] = value;

}

void pop() {

        if (top == -1) {

                printf("Stack Underflow\n");

                return;

        }

        top--;

}

int peek() {

        if (top == -1) {
```

```
        printf("Stack is empty\n");
        return -1;
    }
    return stack[top];
}
int main() {
    push(10);
    push(20);
    push(30);
    printf("Top element: %d\n", peek());
    pop();
    printf("Top element after popping: %d\n", peek());
    return 0;
}
```

output :

Top element: 30

Top element after popping: 20

2 . write a c programming of stack using linked list .

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* top = NULL;
```

```
void push(int value) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = value;
        newNode->next = top;
        top = newNode;
}
void pop() {
        if (top == NULL) {
                printf("Stack is empty\n");
                return;
        }
        struct Node* temp = top;
        top = top->next;
        free(temp);
}
int peek() {
        if (top == NULL) {
                printf("Stack is empty\n");
                return -1;
        }
        return top->data;
}
void display() {
        struct Node* temp = top;
        if (temp == NULL) {
```

```
        printf("Stack is empty\n");
        return;
    }
    printf("Stack elements: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
        printf("\n");
    }
    int main() {
        push(10);
        push(20);
        push(30);
        display();
        printf("Top element: %d\n", peek());
        pop();
        display();
         printf("Top element after pop: %d\n", peek());
      return 0;
}
```

output :

Stack elements: 30 20 10

Top element: 30

Stack elements: 20 10

Top element after pop: 20