Aim

Implement stack operations: PUSH, POP, and PEEK using an array.

Algorithm

1. Initialize stack array and top = -1.
2. For PUSH: check overflow, then increment top and store element.
3. For POP: check underflow, then return element at top and decrement top.
4. For PEEK: show element at top without removing.
5. Menu-driven loop until exit.

Code

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

int stack[MAX], top = -1;

void push(int x) {
    if (top == MAX-1) { printf("Stack Overflow\n"); return; }
    stack[++top] = x;
}

void pop() {
    if (top == -1) { printf("Stack Underflow\n"); return; }
    printf("Popped: %d\n", stack[top--]);
}

void peek() {
    if (top == -1) { printf("Stack is empty\n"); return; }
    printf("Top element: %d\n", stack[top]);
}

void display() {
    if (top == -1) { printf("Stack is empty\n"); return; }
```

```
    for (int i = top; i >= 0; i--) printf("%d ", stack[i]);
    printf("\n");
}

int main() {
    int choice, val;
    do {
        printf("\n1.PUSH 2.POP 3.PEEK 4.DISPLAY 5.EXIT\nChoice: ");
        if (scanf("%d", &choice)!=1) return 0;
        switch(choice) {
            case 1: printf("Enter value: "); scanf("%d",&val);
push(val); break;
            case 2: pop(); break;
            case 3: peek(); break;
            case 4: display(); break;
        }
    } while(choice != 5);
    return 0;
}
```

Sample Output

```
1.PUSH 2.POP 3.PEEK 4.DISPLAY 5.EXIT
Choice: Choice: 1
Enter value: 10

Choice: 1
Enter value: 20

Choice: 4
20 10

Choice: 3
Top element: 20

Choice: 2
Popped: 20



=== Code Execution Successful ===
```

Result

Stack operations (PUSH/POP/PEEK) implemented successfully.