Program 1

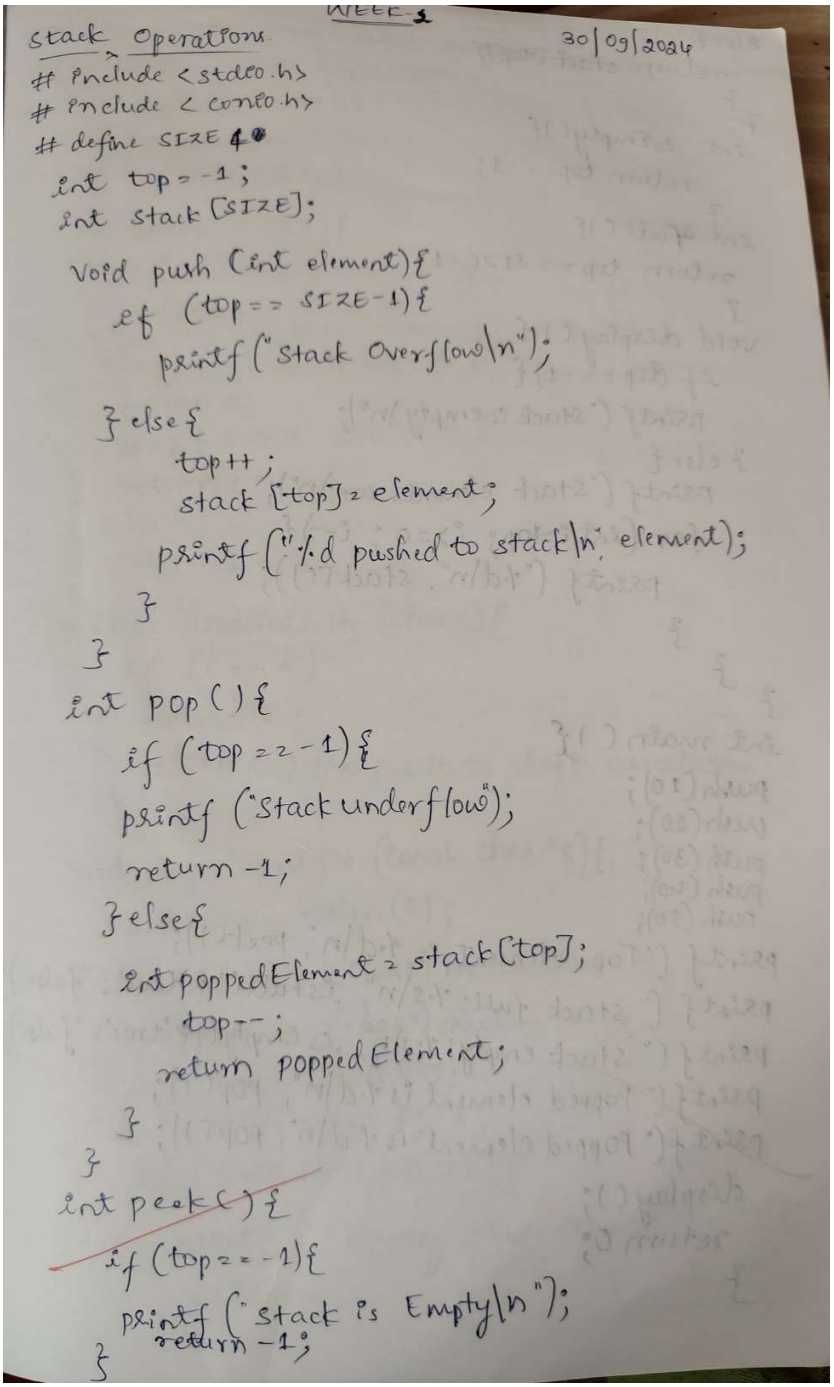
Write a program to simulate the working of stack using an array with the following: a) Push

b) Pop

c) Display

The program should print appropriate messages for stack overflow, stack underflow

Observation:



```
WEEK 2
Stack Operations                                30/09/2024
#include <stdio.h>
#include <conio.h>
#define SIZE 4
int top = -1;
int stack[SIZE];

void push(int element){
    if (top == SIZE-1){
        printf("Stack Overflow\n");
    }else{
        top++;
        stack[top] = element;
        printf("%d pushed to stack\n", element);
    }
}
int pop(){
    if (top == -1){
    printf("Stack underflow");
    return -1;
    }else{
    int poppedElement = stack[top];
        top--;
        return poppedElement;
    }
}
int peek(){
    if (top == -1){
    printf("Stack is Empty\n");
        return -1;
}
```

```
else{
    return stack[top];
  }
}
int isEmpty(){
    return top==-1;
  }
int isfull(){
  return top==SIZE-1;
  }
void display(){
    if(top==-1){
    printf("stack is empty\n");
  }else{
    printf("stack elements are:\n");
    for(int i=top; i>=0; i--){
        printf("%d\n", stack[i]);
      }
  }
}
int main(){
 push(10);
 push(20);
 push(30);
 push(40);
 push(50);
printf("Top element is %d\n", peek());
printf("stack full: %s\n", isFull()?"true":"false");
printf("stack empty: %s\n", isEmpty()?"true":"false");
printf("Popped element is %d\n", pop());
printf("Popped element is %d\n", pop());
 display();
  return 0;
}
```

Code:

```c
#include <stdio.h>
#include <stdlib.h>

int top = -1;


int isEmpty(int arr[]) {
    return top == -1;
}

int isFull(int arr[], int limit) {
    return top == limit - 1;
}

int Top(int arr[]) {
    if (isEmpty(arr)) {
        printf("Stack is empty.\n");
        return -1;
    }
    return arr[top];
}

void Display(int arr[]) {
    if (isEmpty(arr)) {
        printf("Stack is empty.\n");
        return;
    }
    printf("Stack elements: ");
    for (int i = top; i >= 0; i--) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}


void Push(int value, int arr[], int limit) {
    if (isFull(arr, limit)) {
        printf("Stack is full.\n");
    } else {
        top++;
        arr[top] = value;
        printf("Pushed %d onto the stack.\n", value);
    }
}

void Pop(int arr[]) {
    if (isEmpty(arr)) {
        printf("Stack is empty.\n");
    } else {
        printf("Popped %d from the stack.\n", arr[top]);
```

```
        top--;
    }
}

int main() {
    int limit;

    printf("Enter the limit of the stack: ");
    scanf("%d", &limit);

    int arr[limit];

    while (1) {
        int choice, value;
        printf("\nStack Operations:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Top\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to push: ");
                scanf("%d", &value);
                Push(value, arr, limit);
                break;
            case 2:
                Pop(arr);
                break;
            case 3:
                printf("Top element is: %d\n", Top(arr));
                break;
            case 4:
                Display(arr);
                break;
            case 5:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```

Output:

```
PS C:\Users\satis>  & 'c:\Users\satis\.vscode\extensions\ms-vscode.cpptools-1.22.11-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-mljl4g2o.ym5' '--stdout=Microsoft-MIEngine-Out-0hcv5fjb.eug' '--stderr=Microsoft-MIEngine-Error-ewbwcvr2.gjn' '--pid=Microsoft-MIEngine-Pid-2xpmhtiz.bpy' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the limit of the stack: 5

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 1
Pushed 1 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 2
Pushed 2 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 3
Pushed 3 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 4
Pushed 4 onto the stack.
```

```
Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 5 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 4
Stack elements: 4 3 2 1

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 4 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 3 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 2 from the stack.
```

```
Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 5
Pushed 5 onto the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 6
Stack is full.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 3
Top element is: 5

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 4
Stack elements: 5 4 3 2 1
```

```
Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Popped 1 from the stack.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 2
Stack is empty.

Stack Operations:
1. Push
2. Pop
3. Top
4. Display
5. Exit
Enter your choice: 5
```