**A PROJECT REPORT**

**Submited by**

**AJINKYA KEDIA**

**(2030331245013)**

**To**

**Dr. Babasaheb Ambedakar Technological Universtiy, lonere in**

**the fulfillment of the requriments for the award of the Degree**

**Of**

**Bachelor of Technology**

**In**

***Computer Engineering***



**Department of Computer Engineering**

**Dr. Babasaheb Ambedakar Technologicl Universtiy, lonere**

**2020-2021**

**ACKNOWLEDGEMENT**

We undersigned hereby declare that the project report “ DATA STRUCTURE ” , submitted for partial fulfillment of the requirements for the award of degree of B. Tech. of Dr. Babasaheb Ambedkar Technological University, Lonere, is a bonafide work done by us under supervision of ”ProfESSOR LAXMAN NETAK Sir”. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. we also declare that we have adhered to ethics of academic honesty and integrity.I express my gratitude to the Department of Computer Engineering for encouraging us through the entire course work.

Place: Lonere

Date:: 16/5/2021

IMPLEMENTATION OF STACK USING ARRAY

STACK :-

1. ITS an abstract data type which contain two operation basically push and pop
2. It follow LIFO data type

array:

* Contionous memory allocation
* Random access to element ,accessing of element position faster
* Array uses basically 4 bit of data
* Multiple data item in one single name

Condition of using an array:

* The size of the stack is pre-set so it cannot increase or decrease.

## Implementation

* **Push(a):** It adds element a on top of the stack. It takes O(1)*O*(1) time as each element is inserted starting from the table of the array; there is no need to shift existing elements to make room for the new element.
* **Pop():** It removes the element on top of the stack. It also takes O(1)*O*(1) time as the top contains the index of the most recently added element.
* **Top():** It returns the element on top of the stack. It takes O(1)*O*(1) time as finding the value stored at a particular index in an array is a constant time operation.

## Algorithm

### 1).Push()

* Check if the stack is full.
* If the stack is full, then display **"Stack overflow"**.
* If the stack is not full, increment **top** to the next location.
* Assign data to the top element.

Algorithm\_push()

:- IF TOP >= SIZE - 1 then // Stack overflow indicating that the stack is full.

TOP = TOP + 1

STACK [TOP] = ELEMENT

### 2.Pop()

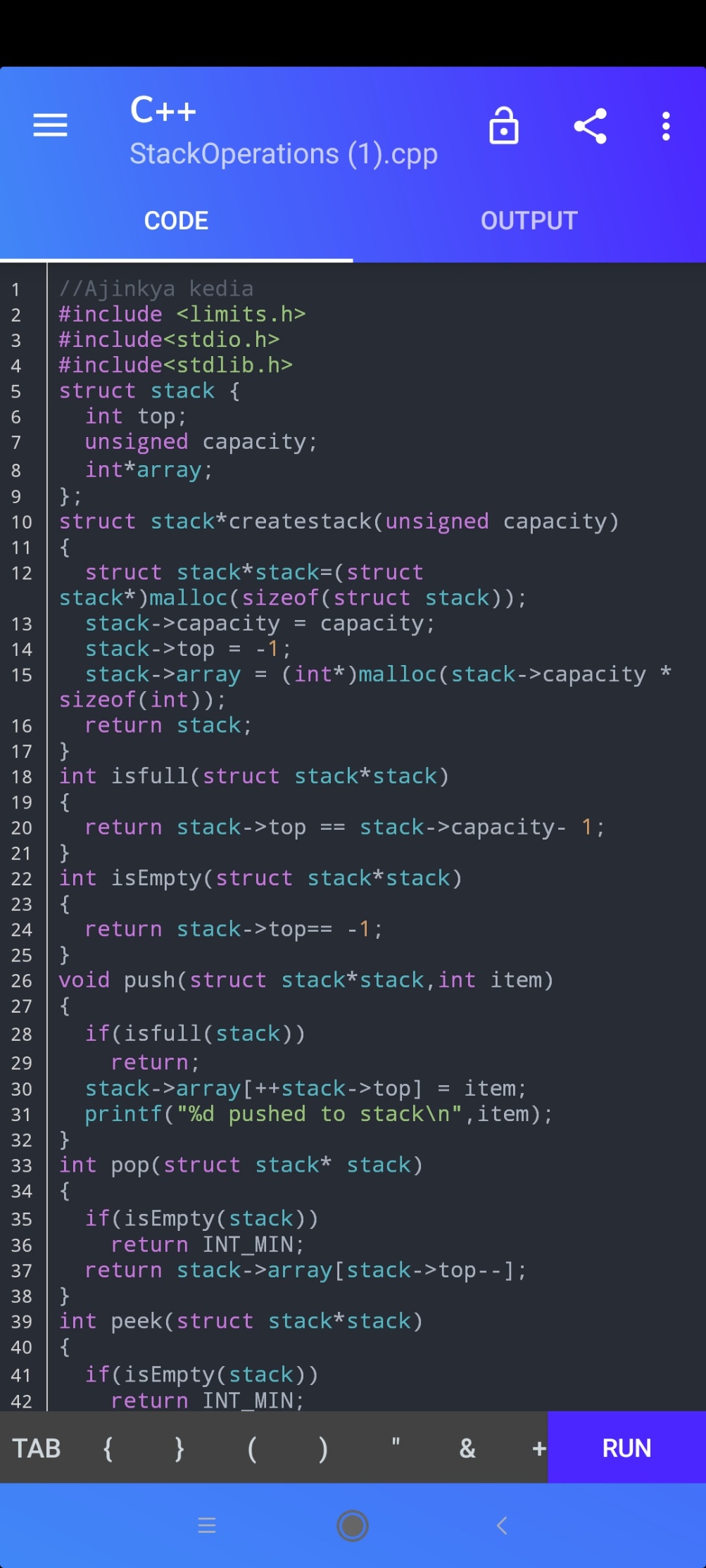
* Check if the stack is empty.
* If the stack is empty, then display **"Stack Underflow"**.
* If the stack is not empty, copy top in a temporary variable.
* Decrement **top** to the previous location.
* Delete the temporary variable. Algorithm\_pop()

:-IF TOP = - 1 then // Stack underflow indicating that the stack is empty.

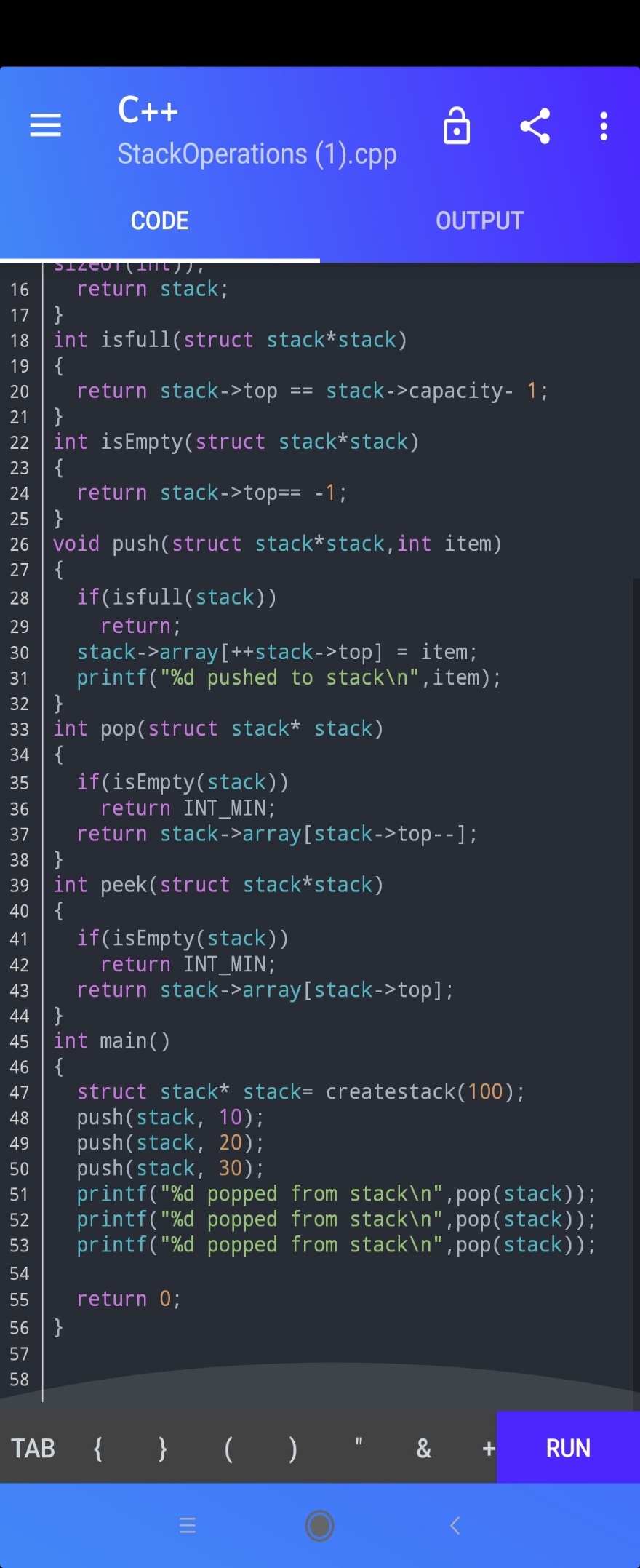
RETURN STACK [TOP] TOP = TOP - 1

# Uses:-

1. Stacks can be used for expression evaluation.
2. Stacks can be used to check parenthesis matching in an expression.
3. Stacks can be used for Conversion from one form of expression to another.
4. Stacks can be used for Memory Management.
5. Stack data structures are used in backtracking problems.



Program :- of stack implementation in array



## Output

