Module 2

Stack and Queues

Agenda

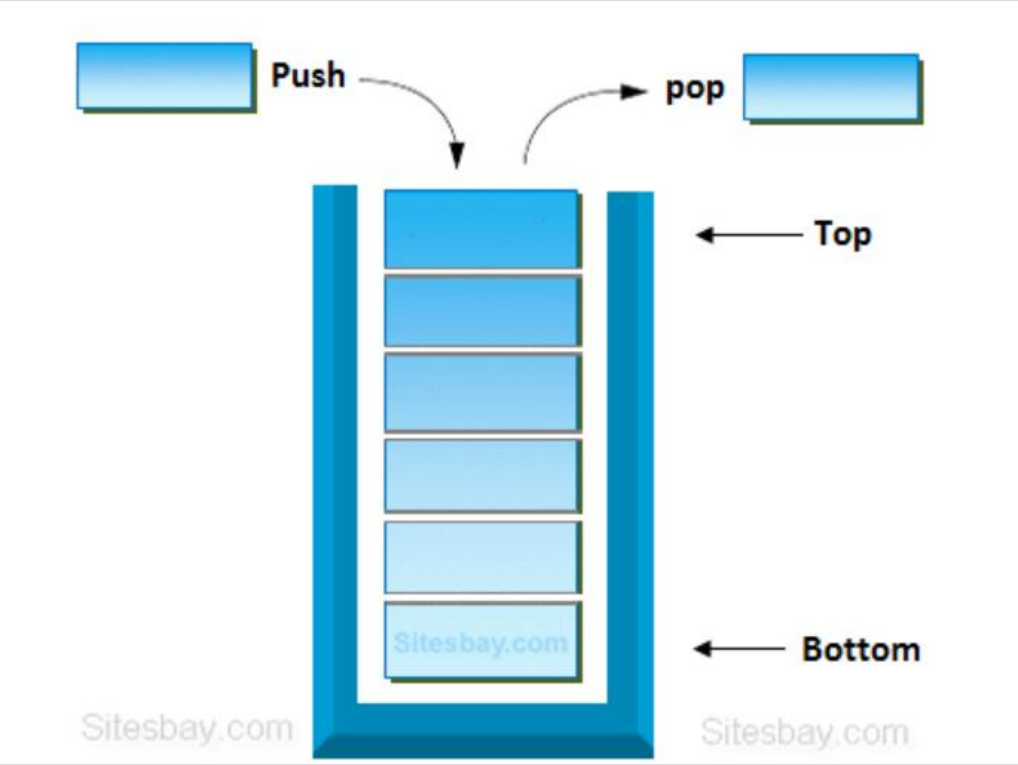
- Introduction
- Operations on Stack
- Implementation of Stack
- Applications of Stack

Defining a Stack

- A stack is a collection of data items that can be accessed at only one end, called top.
- Items can be inserted and deleted in a stack only at the top.
- The last item inserted in a stack is the first one to be deleted.
- Therefore, a stack is called a Last-In-First-Out (LIFO) data structure.

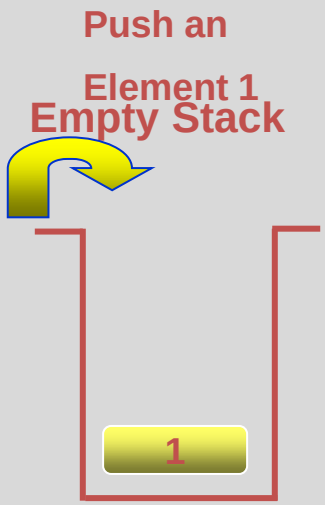
Identifying the Operations on Stacks

- **PUSH**: It is the process of inserting a new element on the top of a stack.



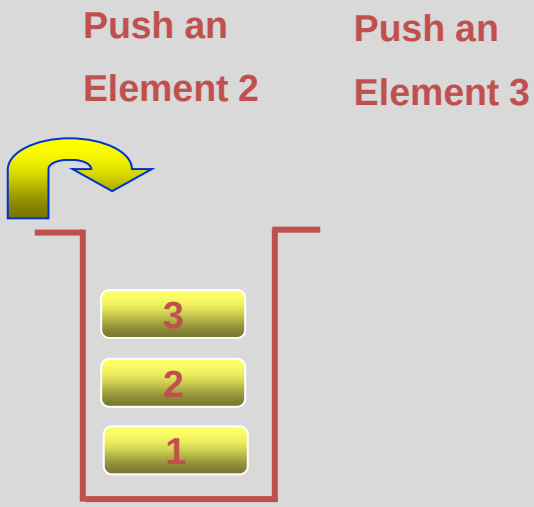
Identifying the Operations on Stacks

- **PUSH**: It is the process of inserting a new element on the top of a stack.
  - There are two basic operations that are performed on stacks:
    - PUSH
    - POP



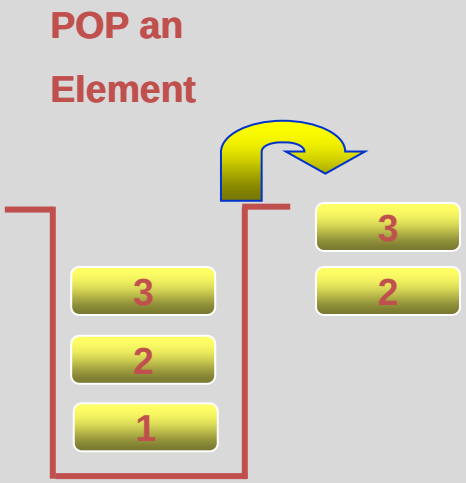
Identifying the Operations on Stacks (Contd.)

- **PUSH**: It is the process of inserting a new element on the top of a stack.

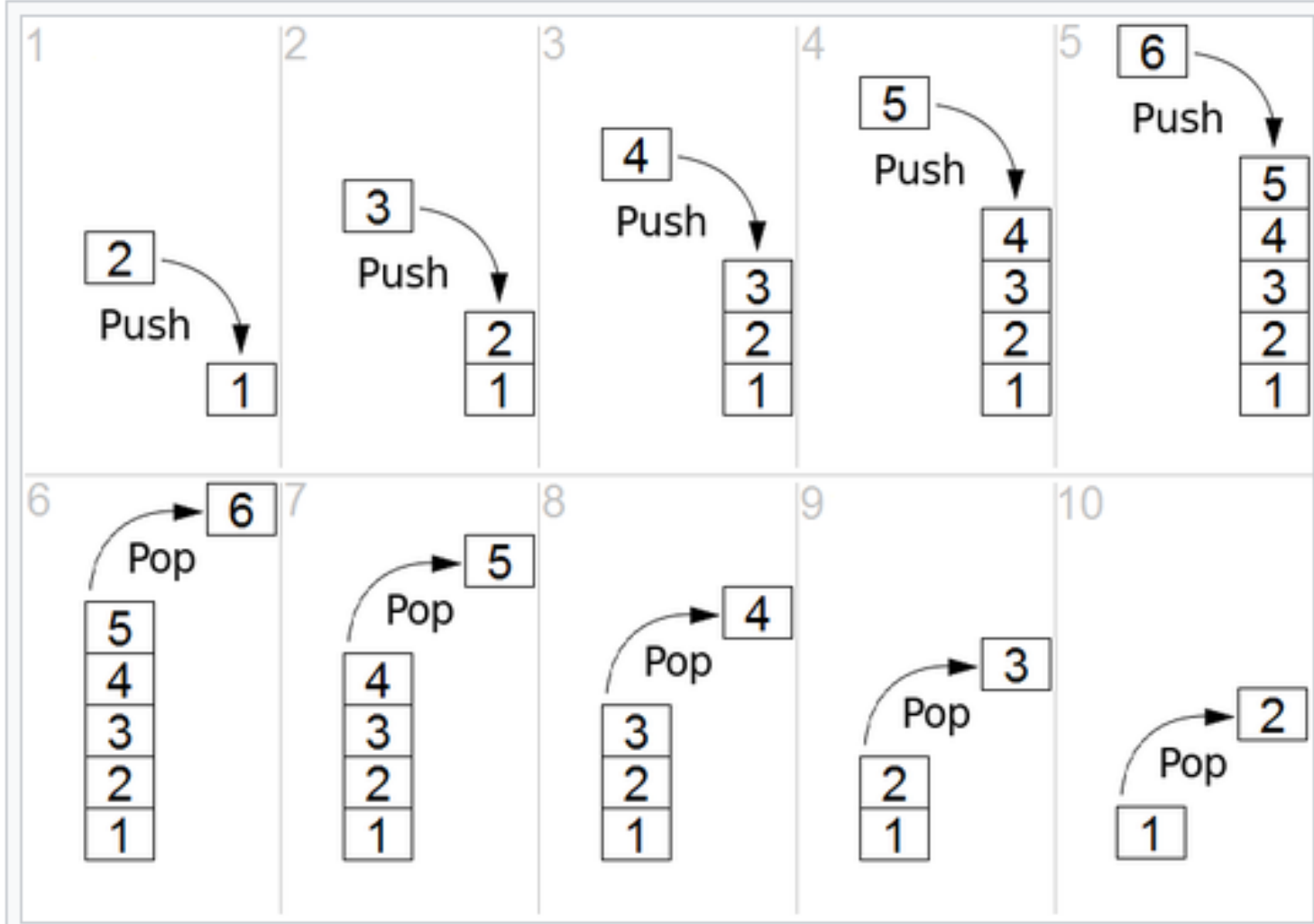


Identifying the Operations on Stacks (Contd.)

- **POP**: It is the process of deleting an element from the top of a stack.

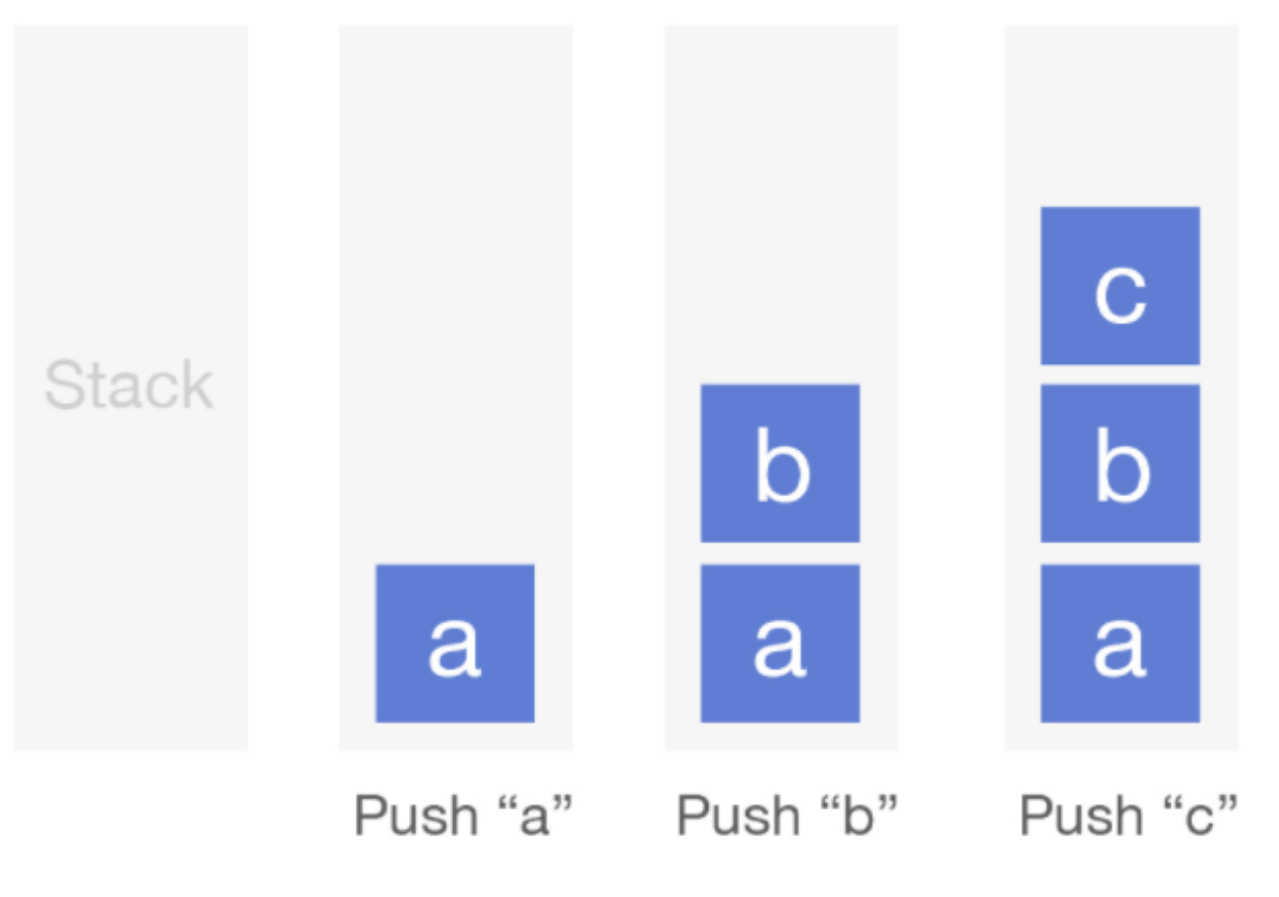


Identifying the Operations on Stacks (Contd.)



Simple representation of a stack runtime with *push* and *pop* operations.

Push Operation

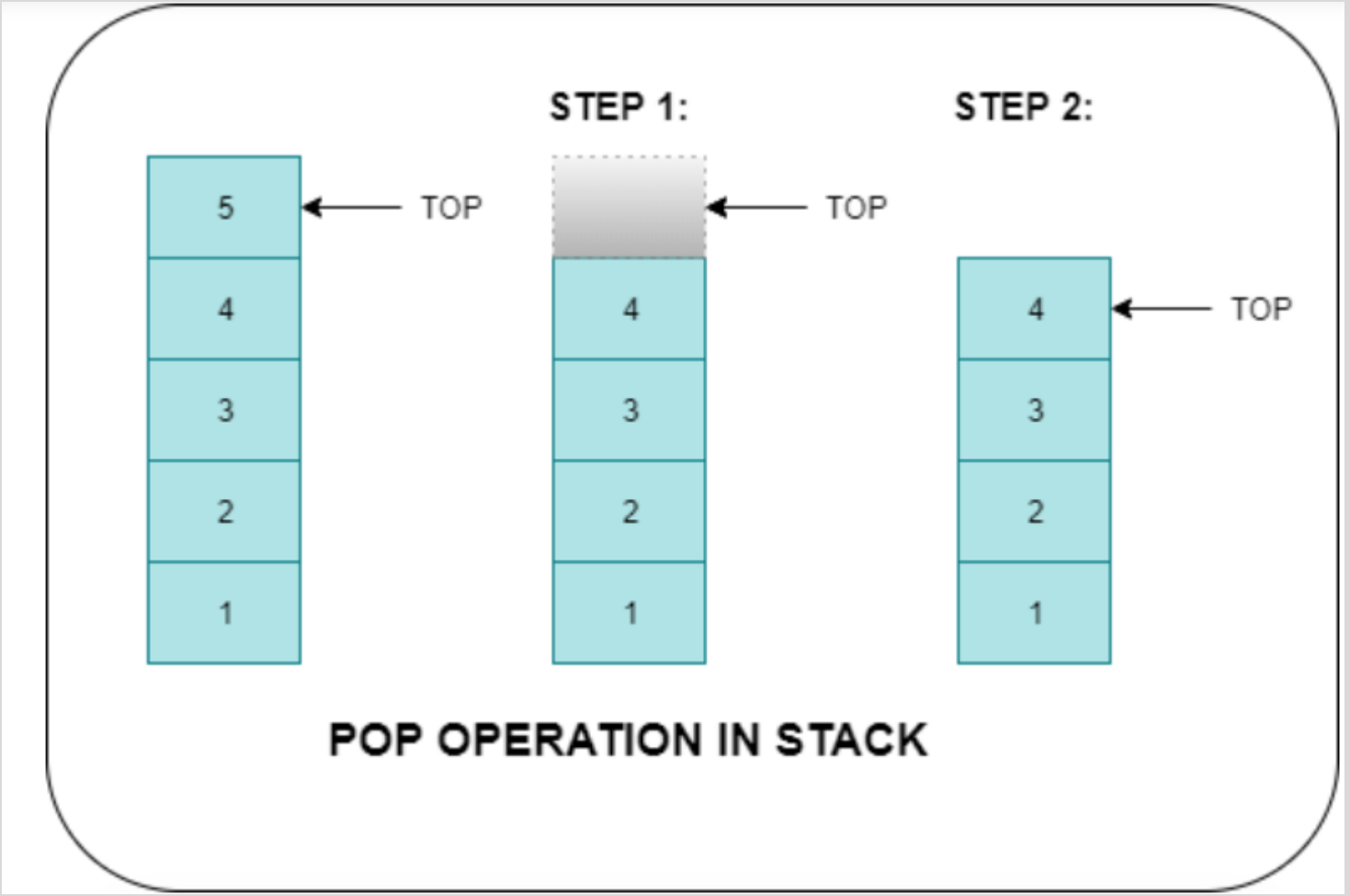


Push Operation

Algorithm for push

1. Initialization, set top=-1
2. Repeat step 3 to 5 until top<Max size-1
3. Read, item
4. Set top=top+1
5. Set stack[top]=item
6. Print "stack overflow"

Pop Operation



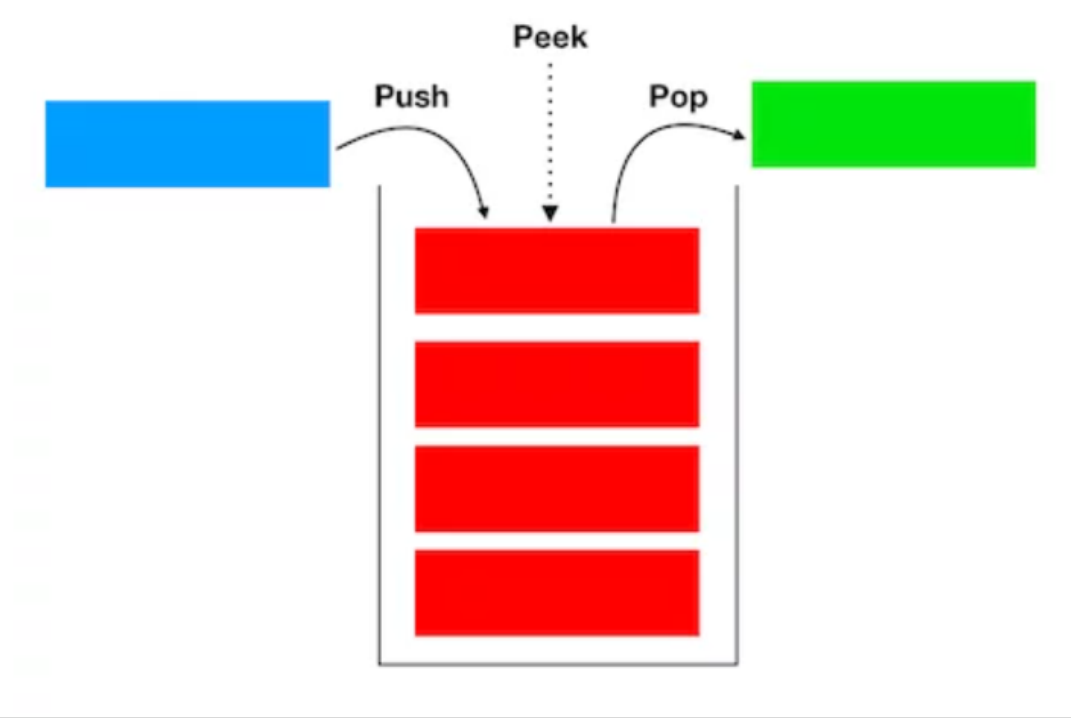
Pop operation

Algorithm for pop

1. Repeated steps 2 to 4 until top>=0
2. Set item=stack[top]
3. Set top=top-1
4. Print "Item deleted"
5. Print "Stack under flow"

Peek/top operation

- Peek/top operation means display the item present at the top of the stack.



Implementing a Stack Using an Array

- A stack is similar to a list in which insertion and deletion is allowed only at one end.
- Therefore, similar to a list, stack can be implemented using both arrays and linked lists.

- To implement a stack using an array:
  - Declare an array:

```
int Stack[5]; // Maximum size needs to be specified in
              // advance
```

  - Declare a variable, top to hold the index of the topmost element in the stacks:

```
int top;
```

  - Initially, when the stack is empty, set:

    top = –1

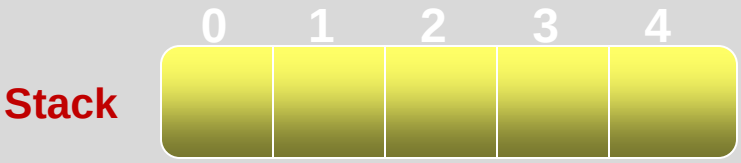
Implementing a Stack Using an Array (contd.)

- Let us now write an algorithm for the PUSH operation.

Initially:

top = – 1

PUSH an element 3

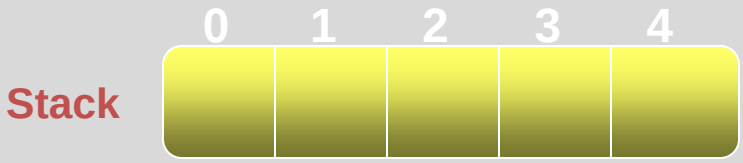


1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

Implementing a Stack Using an Array (Contd.)

top = – 1

PUSH an element 3

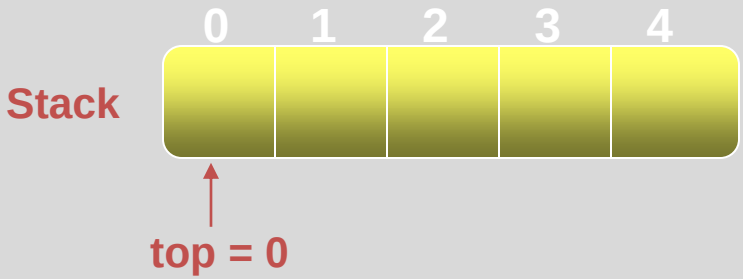


1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

Implementing a Stack Using an Array (Contd.)

top = 0

PUSH an element 3

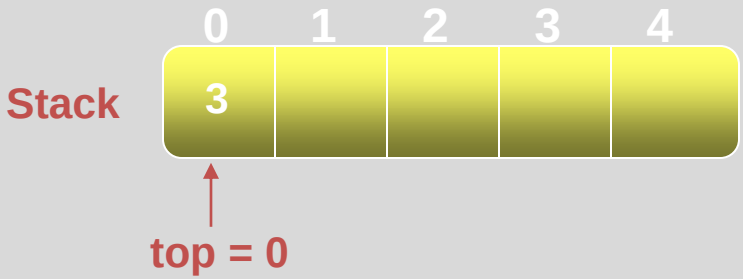


1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 3

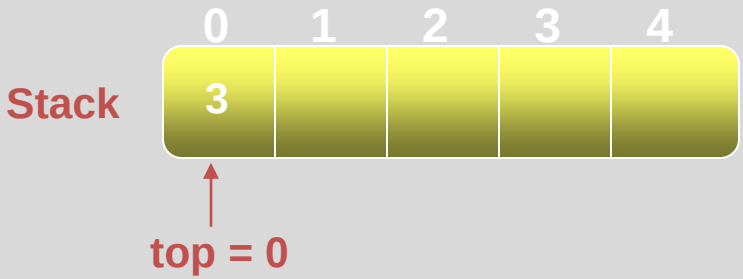


Item pushed

Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 8

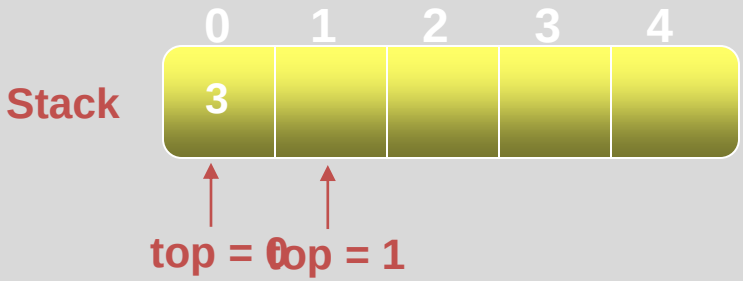


Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.

2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

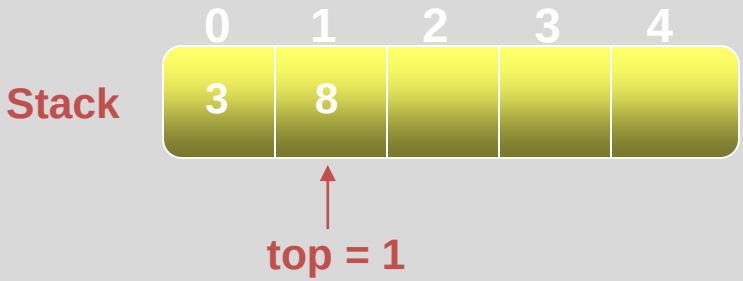
PUSH an element 8



Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 8

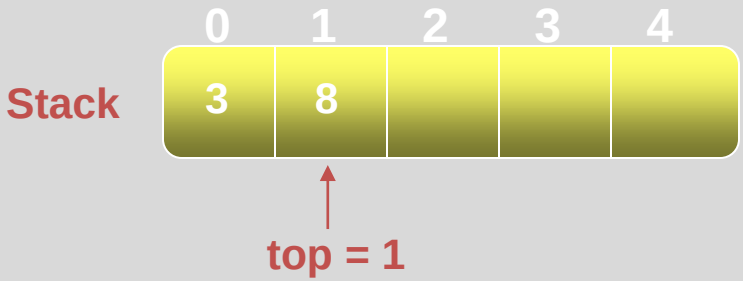


Item pushed

Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 5

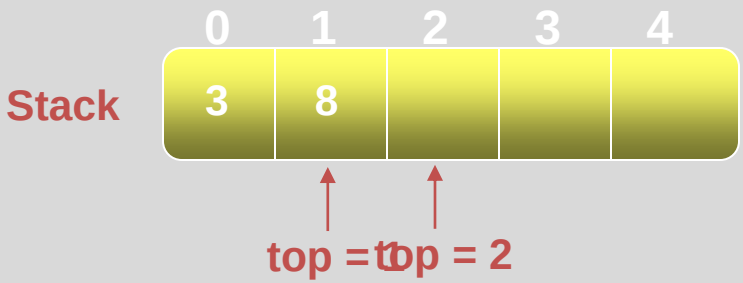


Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.

2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

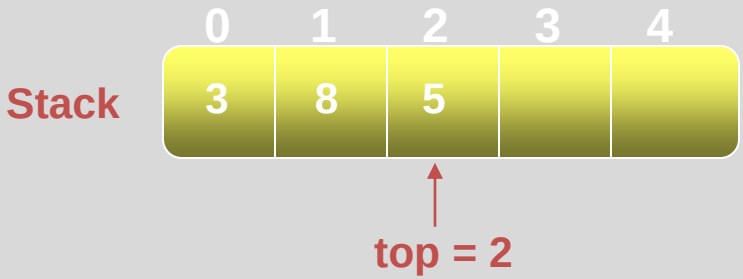
PUSH an element 5



Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 5

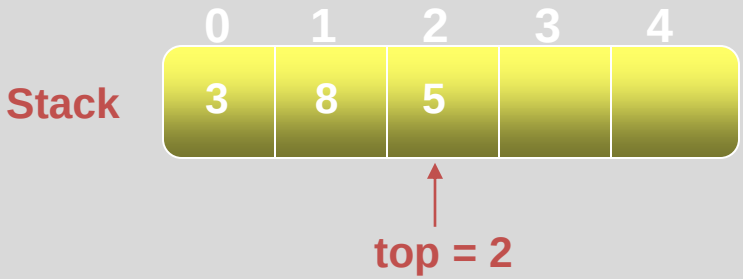


Item pushed

Implementing a Stack Using an Array (Contd.)

1. Increment top by 1
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 1

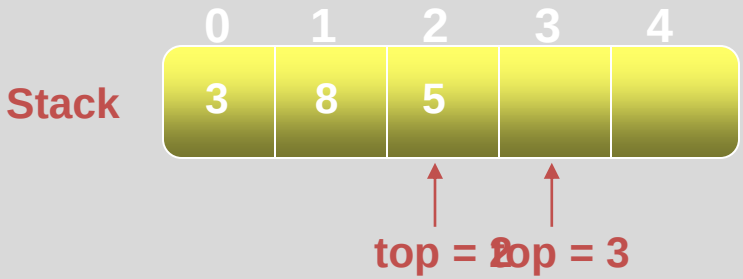


Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.

2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

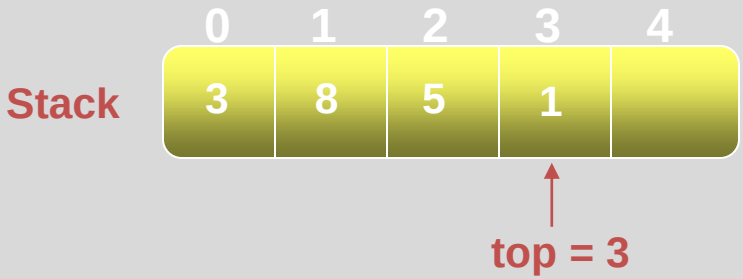
PUSH an element 1



Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 1

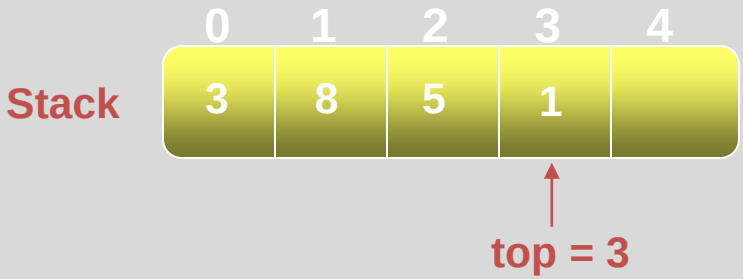


Item pushed

Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

PUSH an element 9

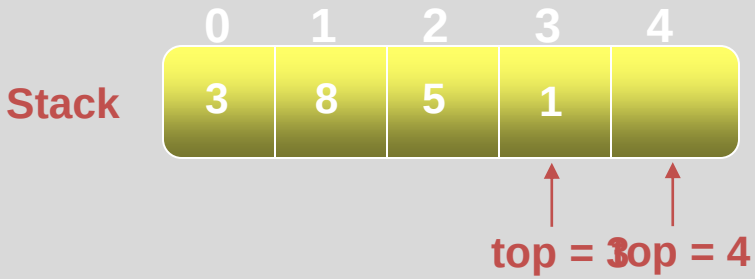


Implementing a Stack Using an Array (Contd.)

1. Increment top by 1

2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

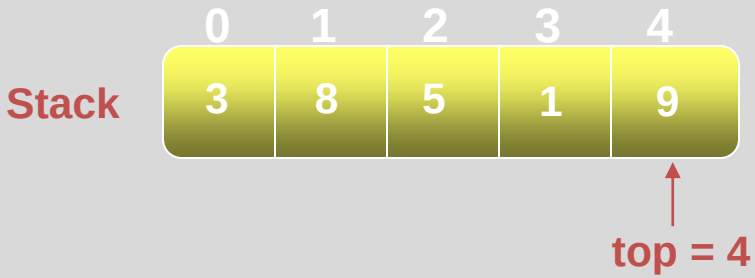
PUSH an element 9



Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

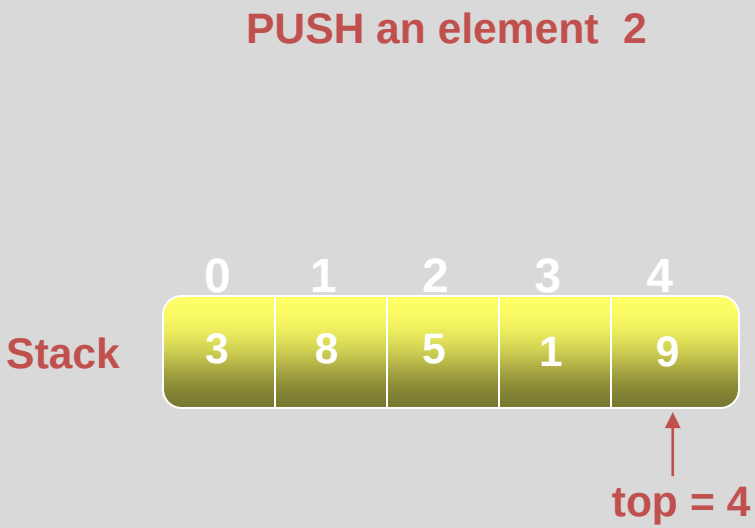
PUSH an element 9



Item pushed

Implementing a Stack Using an Array (Contd.)

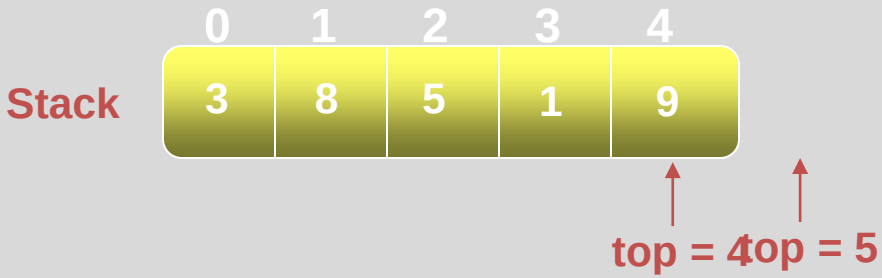
1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.



Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

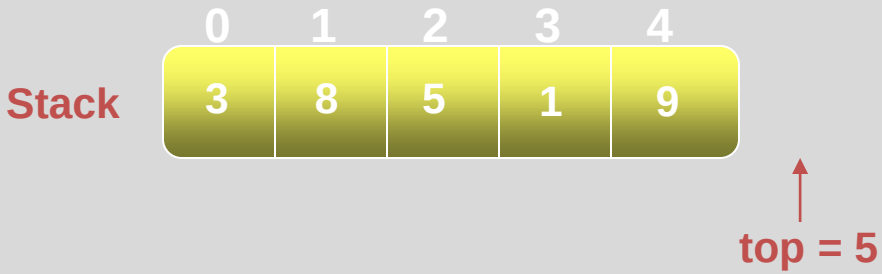
PUSH an element 2



Implementing a Stack Using an Array (Contd.)

1. Increment top by 1.
2. Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.

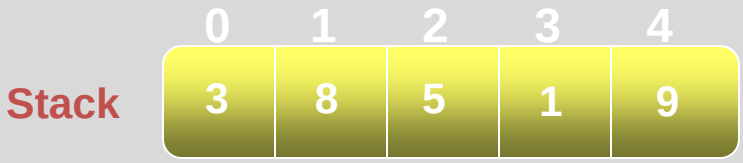
PUSH an element 2



Stack overflow

Implementing a Stack Using an Array (Contd.)

- The stack has been implemented in an array of size 5.
- Therefore, you cannot store more than 5 elements in the stack.
- To avoid the stack overflow, you need to check for the stack full condition before pushing an element into the stack.
- Let us modify the algorithm to check for this condition.



- Increment top by 1.
- Store the value to be pushed at index top in the array. Top now contains the index of the topmost element.
- If top = MAX – 1:
  - Display “Stack Full”
  - Exit
- Increment top by 1
- Store the value to be pushed at index top in the array