# Linked List Stacks

Week 7

Lectures 1 and 2

- In this session, you will learn to:
  - Linked List implementation of Stacks

## Inserting an Element in a Linked Queue

- Write an algorithm to implement insert operation in a linked STACK.

# Inserting an Element in a Linked Queue (Contd.)

- Algorithm to insert an element in a linked queue.

Request number generated **3**

**TOP = NULL**

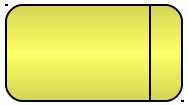
- Suppose initially, the stack is empty.

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to
   the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

**TOP = NULL**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

**TOP = NULL**


1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.

. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

**TOP = NULL**


1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.

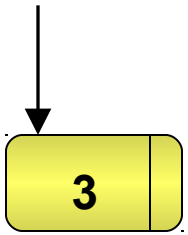
. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

**TOP = NULL**

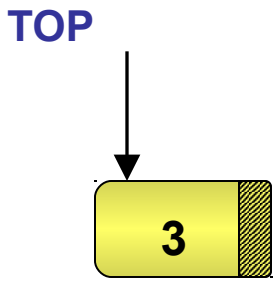
**TOP**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

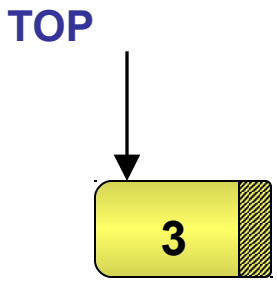



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

**Insert operation complete**

# Inserting an Element in a Linked Queue (Contd.)

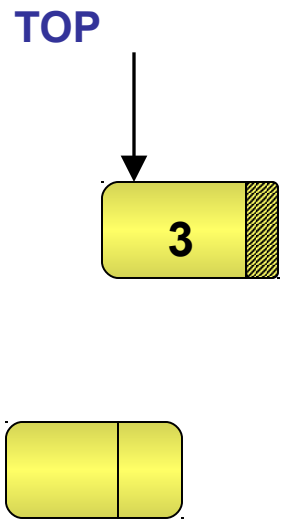
Request number generated **5**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

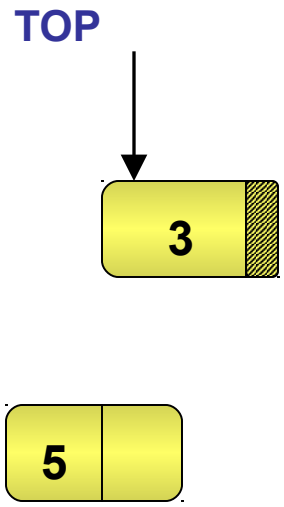
Request number generated **5**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

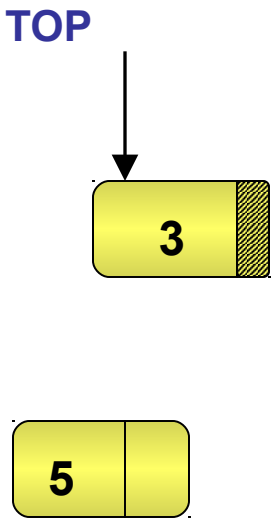
Request number generated **5**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

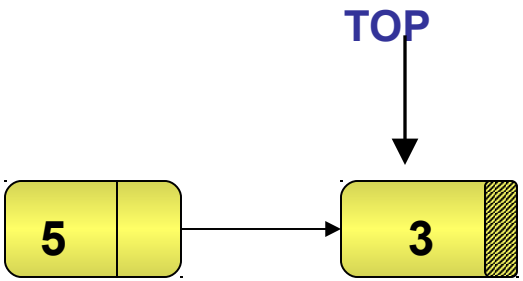
Request number generated **5**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

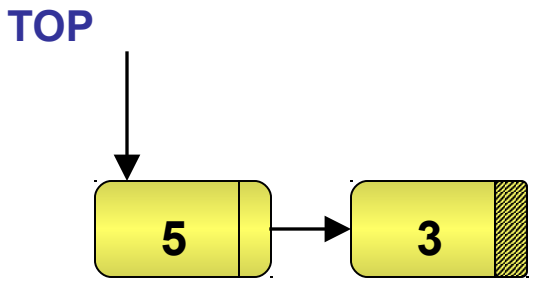
Request number generated **5**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

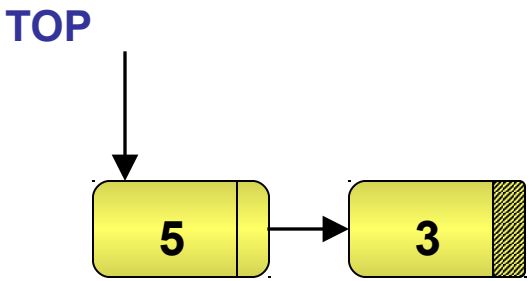
Request number generated **5**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

# Inserting an Element in a Linked Queue (Contd.)

Request number generated **5**




1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If the stack is empty, execute the following steps:
   a. Make TOP point to the new node
   b. Make the next field of new node point to the NULL.
   c. Exit
4. Make the next field of the new node point to the first node in the list.
5. Make TOP, point to the new node.

**Insert operation complete**

# ALGORITHM TO IMPLEMENT INSERT OPERATION

TOP = NULL

Algorithm Insert()

{

1. Create node [(new1=(struct node*) malloc(sizeof(struct node))]
2. Enter data [new1 -> info = data]
3. If (TOP == NULL)

    3.1 new1 -> next = NULL

    3.2 TOP = new1

else

    3.1 new1 -> next = TOP

    3.2 TOP = new1

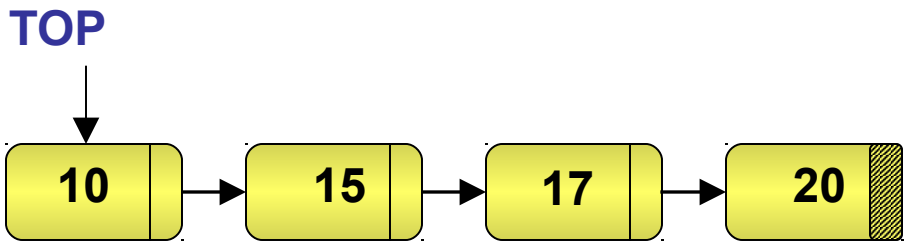
}

## Deleting a Node From the Beginning of the List

- Write an algorithm to implement insert operation in a linked STACK.

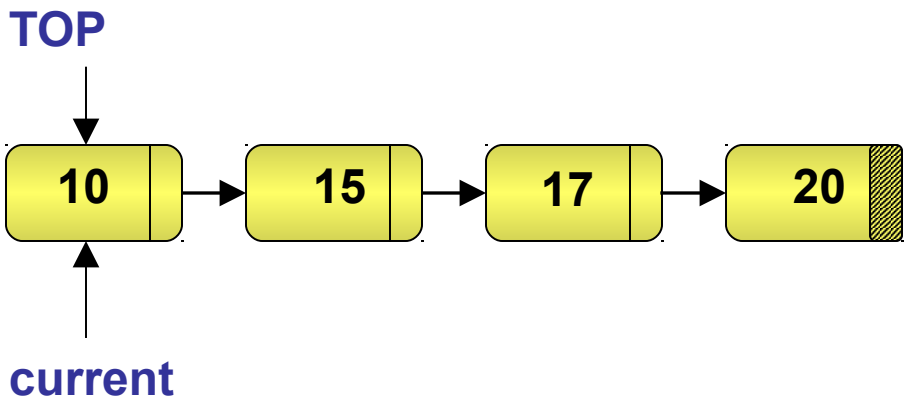
# Deleting a Node From the Beginning of the List (Contd.)

- Algorithm to delete An element




1. Mark the first node in the list as current.
2. Make TOP point to the next node in its sequence.
3. Release the memory for the node marked as current.

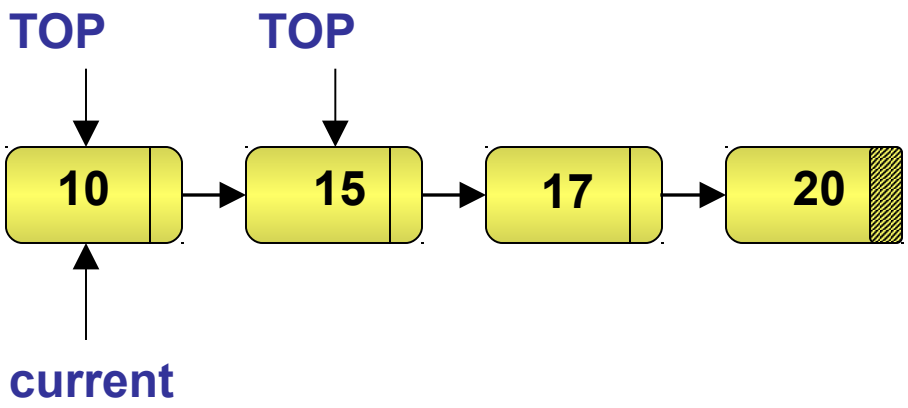
# Deleting a Node From the Beginning of a Linked List (Contd.)




1. Mark the first node in the list as current.
2. Make TOP point to the next node in its sequence.
3. Release the memory for the node marked as current.

**current = TOP**

# Deleting a Node From the Beginning of a Linked List (Contd.)



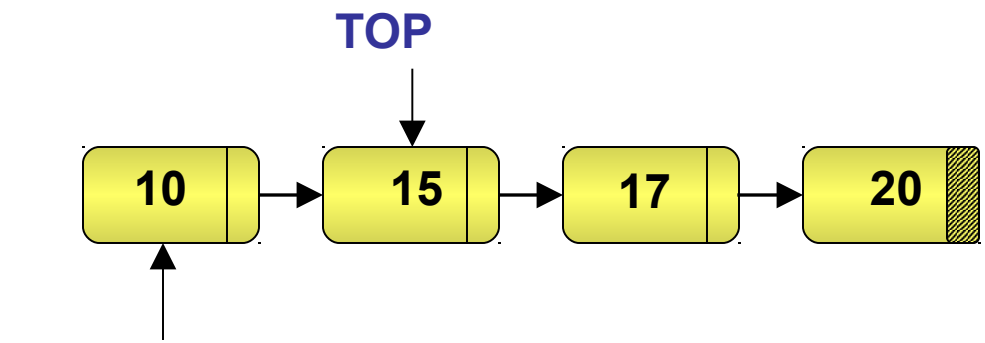

1. Mark the first node in the list as current.
2. Make TOP point to the next node in its sequence.
3. Release the memory for the node marked as current.

**current = TOP**

**TOP = TOP -> next**

# Deleting a Node From the Beginning of a Linked List (Contd.)

1. Mark the first node in the list as current.
2. Make TOP point to the next node in its sequence.
3. Release the memory for the node marked as current.




Memory released

current = TOP

TOP = TOP -> next

Delete operation complete

# ALGORITHM TO IMPLEMENT DELETION OPERATION

```
Algorithm Delete()
{
1. If (TOP == NULL)
        1.1 Print "underflow“
   else
        1.1 Current = TOP
        1.2 TOP = TOP -> next
        1.3 Current -> next = NULL
        1.4 Release the memory [ free (Current) ]

}
```