Singly Linked Lists

Objectives

- In this session, you will learn to:
  - Identify the features of linked lists
  - Implement a singly-linked list
    - Insertion in a linked List
    - Deletion from a Linked List
    - Traversing a Linked list

Linked List

- Suppose you have to write an algorithm to generate and store all prime numbers between 1 and 10,00,000 and display them.
- How will you solve this problem?

Linked List (Contd.)

- Consider the following algorithm, which uses an array to solve this problem:
  1. Set I = 0
  2. Repeat step 3 varying N from 2 to 1000000
  3. If N is a prime number
     a. Set A[I] = N // If N is prime store it in an array
     b. I = I + 1
  4. Repeat step 5 varying J from 0 to I-1
  5. Display A[J] // Display the prime numbers
     // stored in the array

Linked List (Contd.)

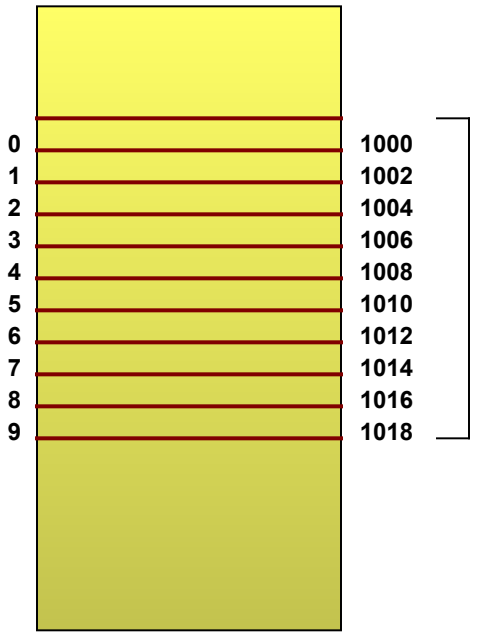
- What is the problem in this algorithm?
  - The number of prime numbers between 1 and 10,00,000 is not known. Since you are using an array to store the prime numbers, you need to declare an array of arbitrarily large size to store the prime numbers.
  - Disadvantages of this approach, suppose you declare an array of size N:
    - If the number of prime numbers between 1 and 10,00,000 is more than N then all the prime numbers cannot be stored.
    - If the number of prime numbers is much less than N, a lot of memory space is wasted.

Linked List (Contd.)

- Thus, you cannot use an array to store a set of elements if you do not know the total number of elements in advance.
- How do you solve this problem?
  - By having some way in which you can allocate memory as and when it is required.

Dynamic Memory Allocation

- When you declare an array, a contiguous block of memory is allocated.
- Let us suppose you declare an array of size 10 to store first 10 prime numbers.
- If you know the address of the first element in the array you can calculate the address of any other elements as shown:
  - Address of the first element + (size of the element × index of the element)

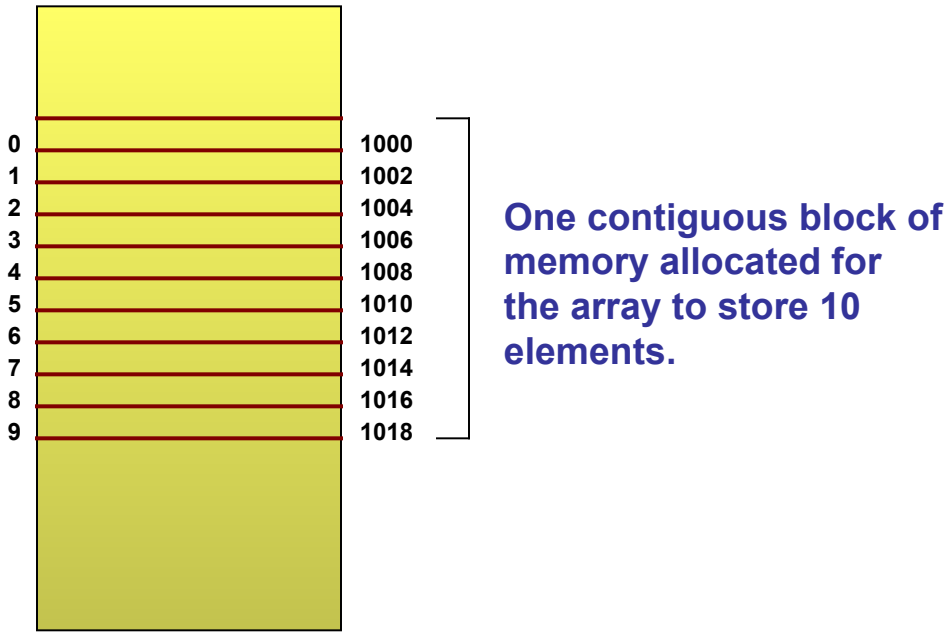


One contiguous block of memory allocated for the array to store 10 elements.

Memory representation

Dynamic Memory Allocation (Contd.)

- When memory is allocated dynamically, a block of memory is assigned arbitrarily from any location in the memory.
- Therefore, unlike arrays, these blocks may not be contiguous and may be spread randomly in the memory.

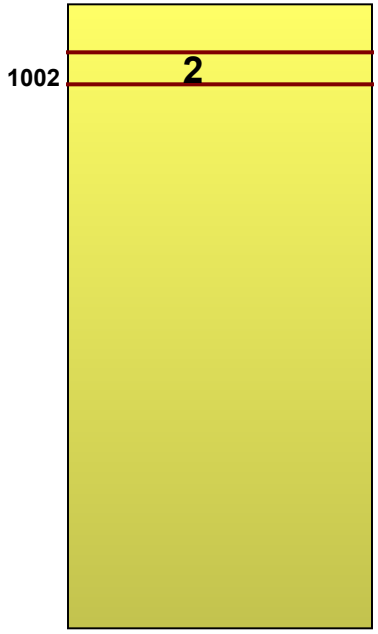


Memory representation

Dynamic Memory Allocation (Contd.)

- Let us see how this happens by allocating memory dynamically for the prime numbers.

Allocate memory for 2



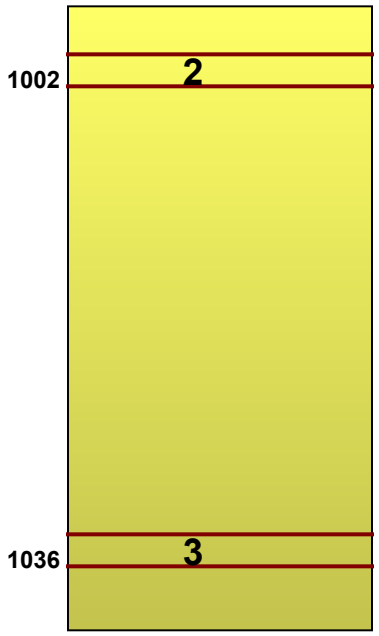
Memory allocated for 2

Memory representation

Dynamic Memory Allocation (Contd.)

- Let us see how this happens.

Allocate memory for 3



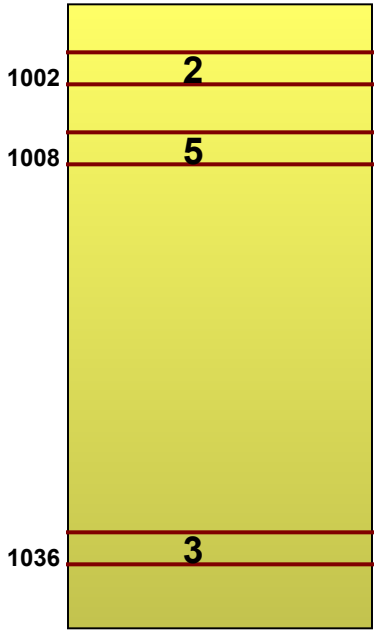
Memory allocated for 3

Memory representation

Dynamic Memory Allocation (Contd.)

- Let us see how this happens.

Allocate memory for 5



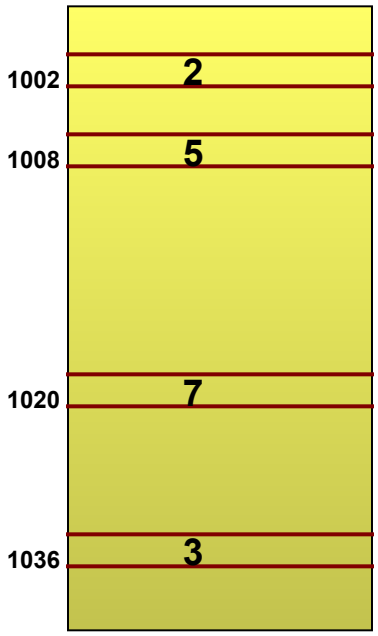
Memory allocated for 5

Memory representation

Dynamic Memory Allocation (Contd.)

- Let us see how this happens.

Allocate memory for 7



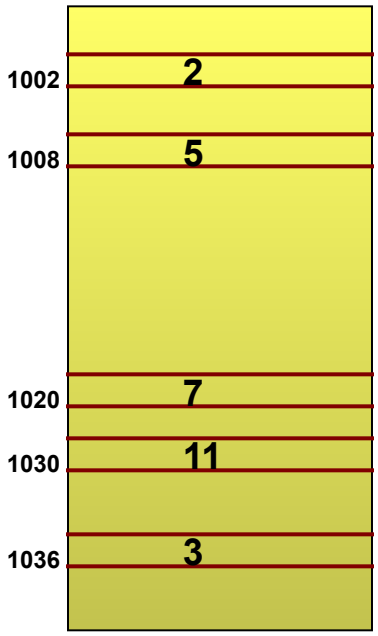
Memory representation

Memory allocated for 7

Dynamic Memory Allocation (Contd.)

- Let us see how this happens.

Allocate memory for 11

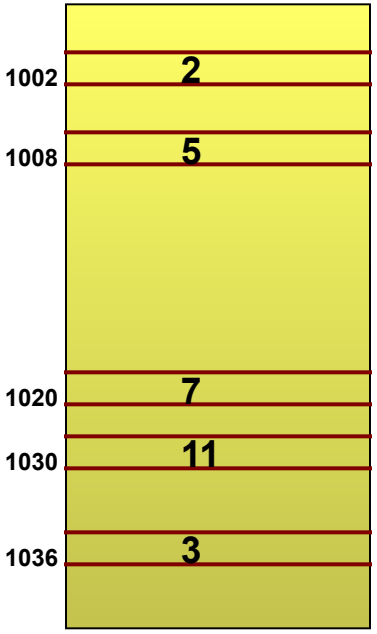


Memory representation

Memory allocated for 11

Dynamic Memory Allocation (Contd.)

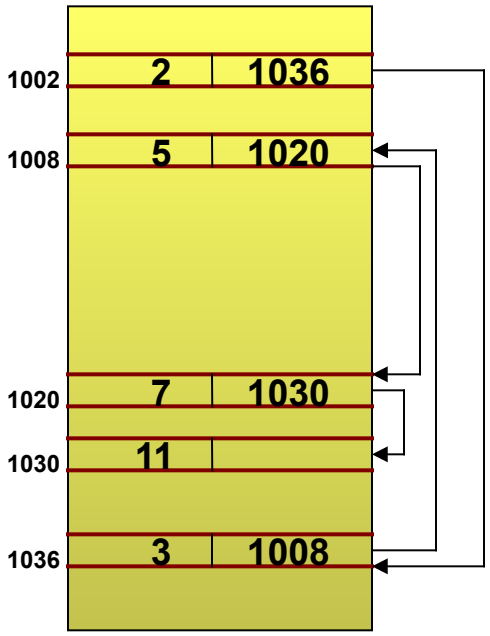
- To access any element, you need to know its address.
  - If you know the address of the first element, you cannot calculate the address of the rest of the elements.
  - This is because, all the elements are spread at random locations in the memory.



Memory representation

Dynamic Memory Allocation (Contd.)

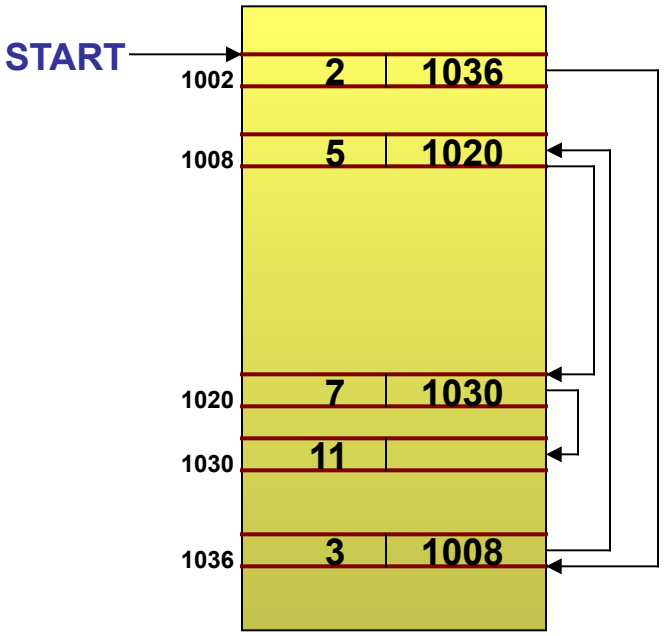
- Therefore, it would be good if every allocated block of memory contains the address of the next block in sequence.
- This gives the list a linked structure where each block is linked to the next block in sequence.



Memory representation

Dynamic Memory Allocation (Contd.)

- An example of a data structure that implements this concept is a linked list.
  - You can declare a variable, START, that stores the address of the first block.
  - You can now begin at START and move through the list by following the links.



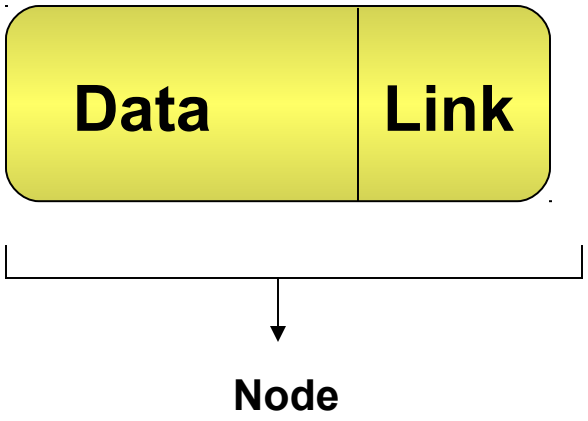
Memory representation

Defining Linked Lists

- Linked list:
  - Is a dynamic data structure.
  - Allows memory to be allocated as and when it is required.
  - Consists of a chain of elements, in which each element is referred to as a node.

Defining Linked Lists (Contd.)

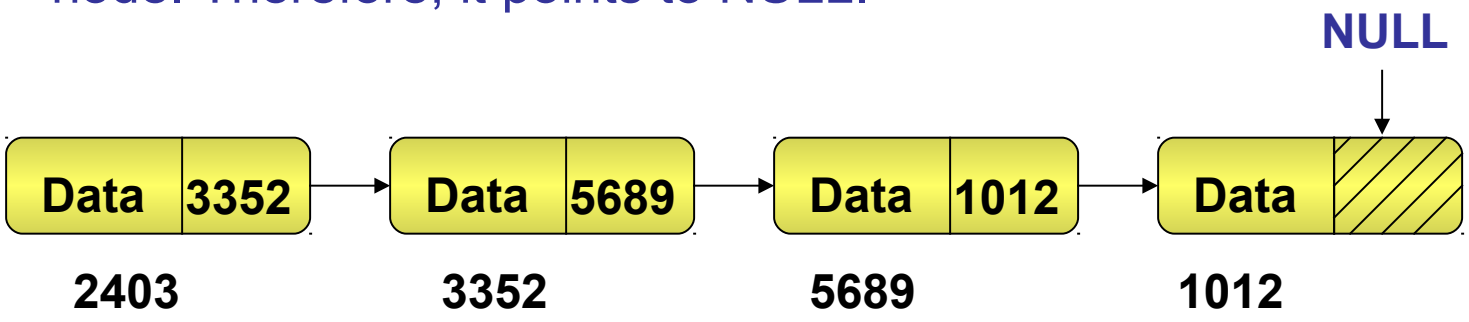
- A node is the basic building block of a linked list.
- A node consists of two parts:
  - **Data**: Refers to the information held by the node
  - **Link**: Holds the address of the next node in the list



Defining Linked Lists (Contd.)

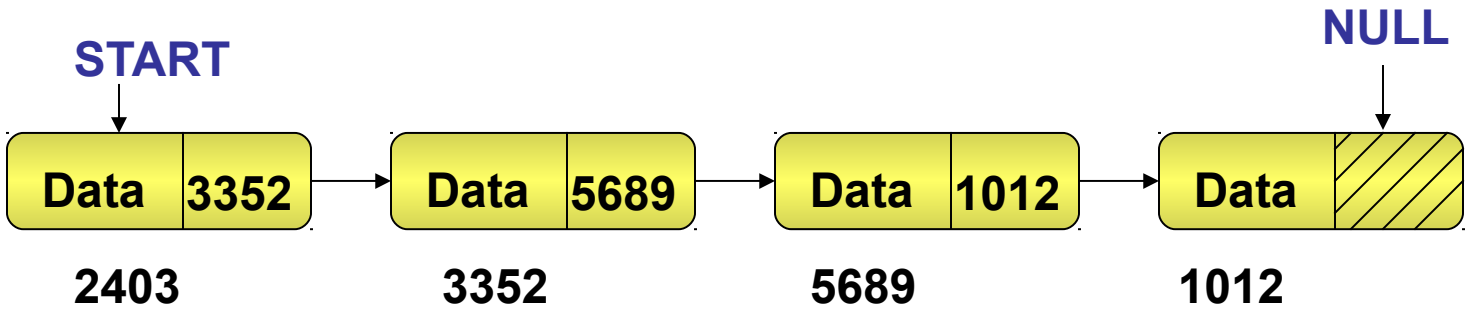
- All the nodes in a linked list are present at arbitrary memory locations.
- Therefore, every node in a linked list has link field that stores the address of the next node in sequence.

- The last node in a linked list does not point to any other node. Therefore, it points to NULL.



Defining Linked Lists (Contd.)

- To keep track of the first node, declare a variable/pointer, START, which always points to the first node.
- When the list is empty, START contains null.



HOW TO CREATE A NODE

```
struct node
{
int  info;
struct node  *next;
};
struct node *new1;
new1=(struct node*) malloc(sizeof(struct node))
new1->info=3;
new1->next=NULL;
```

Inserting a Node in a Singly-Linked List (Contd.)

- A new node can be inserted at any of the following positions in the list:
  - Beginning of the list
  - End of the List
  - At specific position
  - In a Sorted Linked List

Inserting a Node a Singly-Linked LIST

- Refer to the given algorithm to insert a node at the end of the linked list.

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

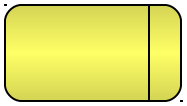
Inserting a Node in a Singly-Linked List (Contd.)

- Consider that the list is initially empty.
- Insert a prime number 2.
  - START = NULL
  - LAST = NULL

1. **Allocate memory for the new node.**
2. **Assign value to the data field of the new node.**
3. **If START is NULL, then (If the list is empty):**
   a. **Make START point to the new node.**
   b. **Make LAST point to the new node.**
   c. **Go to step 6.**
4. **Make the next field of LAST point to the new node.**
5. **Mark the new node as LAST.**
6. **Make the next field of the new node point to NULL.**

Inserting a Node in a Singly-Linked List (Contd.)

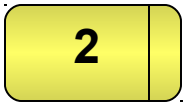
- START = NULL
- Insert a prime number 2.



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)

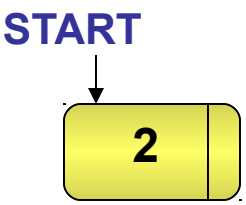
- START = NULL
- LAST = NULL



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)

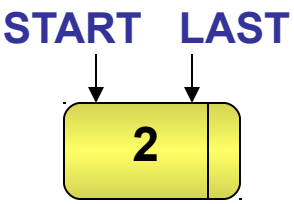
- START = NULL
- LAST = NULL



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

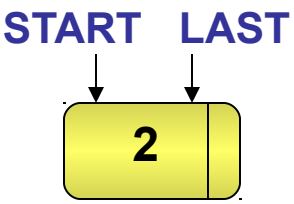
Inserting a Node in a Singly-Linked List (Contd.)

- LAST = NULL



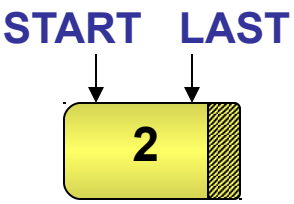
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)

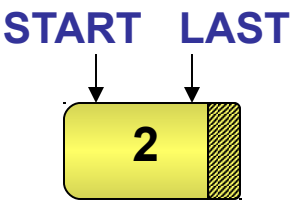


Insertion complete

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)

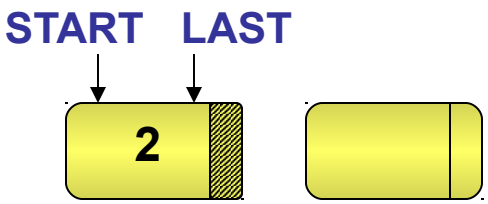
- Insert a prime number 3.



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

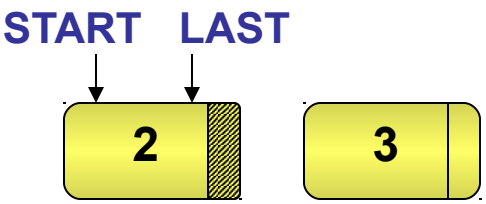
Inserting a Node in a Singly-Linked List (Contd.)

- Insert a prime number 3.



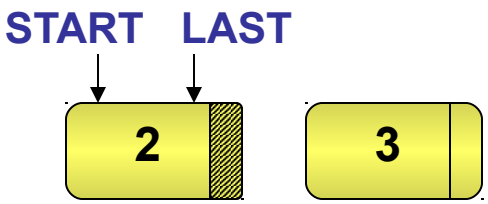
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



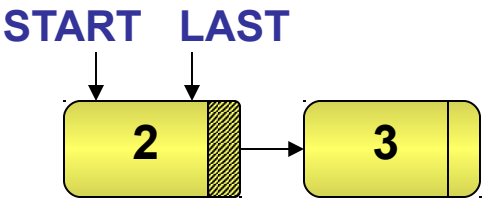
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



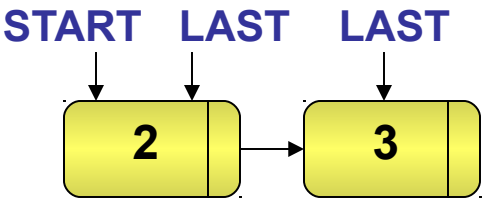
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



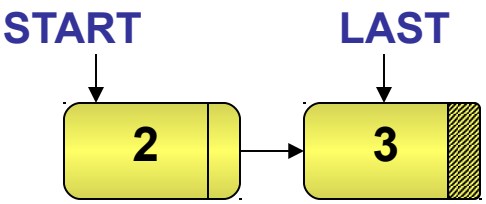
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)

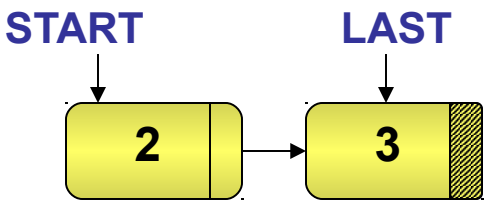


Insertion complete

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)

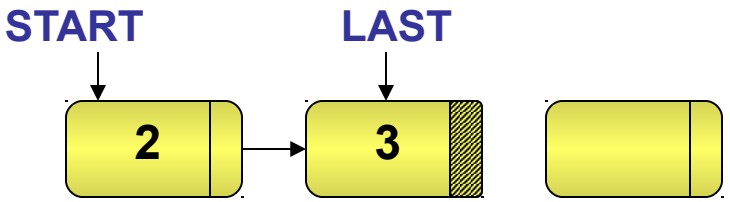
- Insert a prime number 5.



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

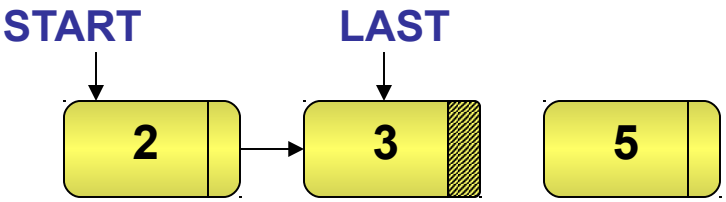
Inserting a Node in a Singly-Linked List (Contd.)

- Insert a prime number 5.



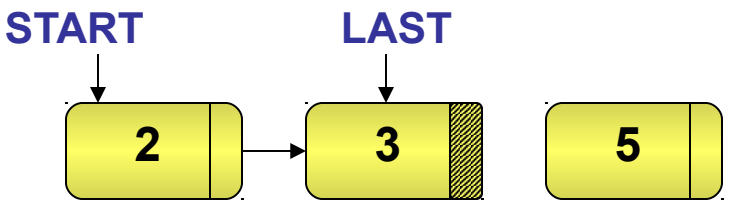
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



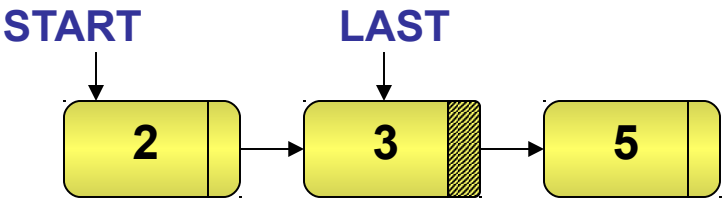
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



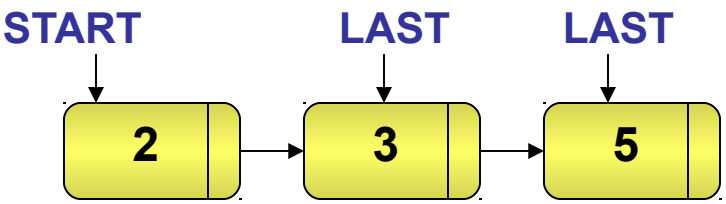
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



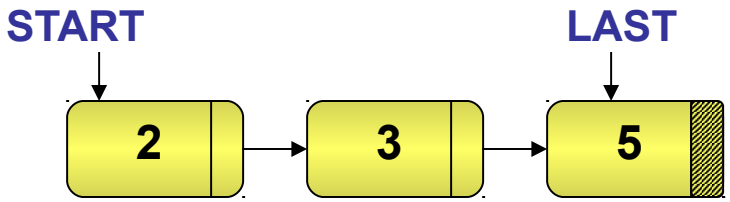
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Inserting a Node in a Singly-Linked List (Contd.)



Insertion complete

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

ALGORITHM TO INSERT NODE AT END

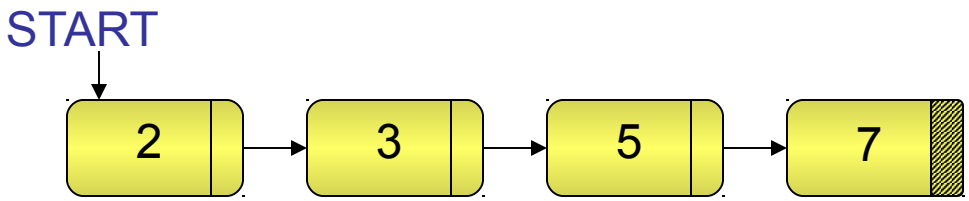
```
Start=NULL
Algorithm InsertAtEnd()
{
1. Create node [(new1 = (struct node*)
     malloc(sizeof(struct node))]
2. Enter data [new1 -> info =d ata]
3.if(Start == NULL)
            3.1 Last=new1
            3.2 Start=new1
   else
            3.1 Last -> next = new1
            3.2 Last = new1
4. Last ->next = NULL

}
```

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If START is NULL, then (If the list is empty):
   a. Make START point to the new node.
   b. Make LAST point to the new node.
   c. Go to step 6.
4. Make the next field of LAST point to the new node.
5. Mark the new node as LAST.
6. Make the next field of the new node point to NULL.

Traversing a Singly-Linked List

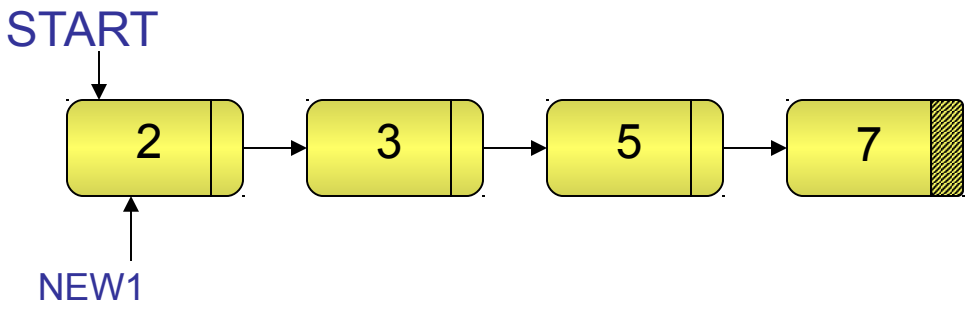
- Write an algorithm to traverse a singly-linked list.
  - Algorithm for traversing a linked list.



1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

Traversing a Singly-Linked List (Contd.)

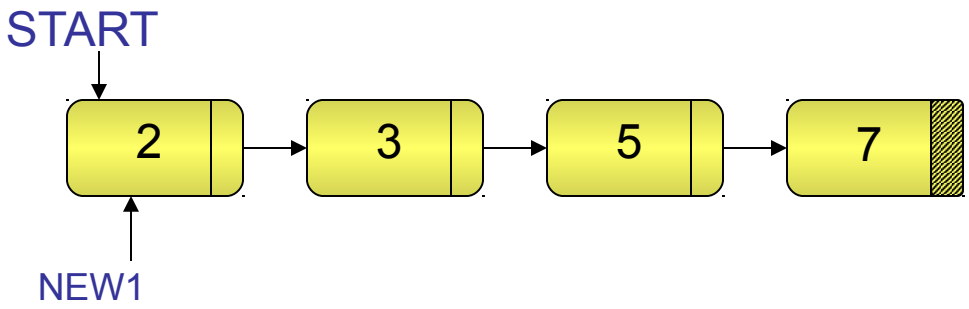
- Refer to the algorithm to display the elements in the linked list.



1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

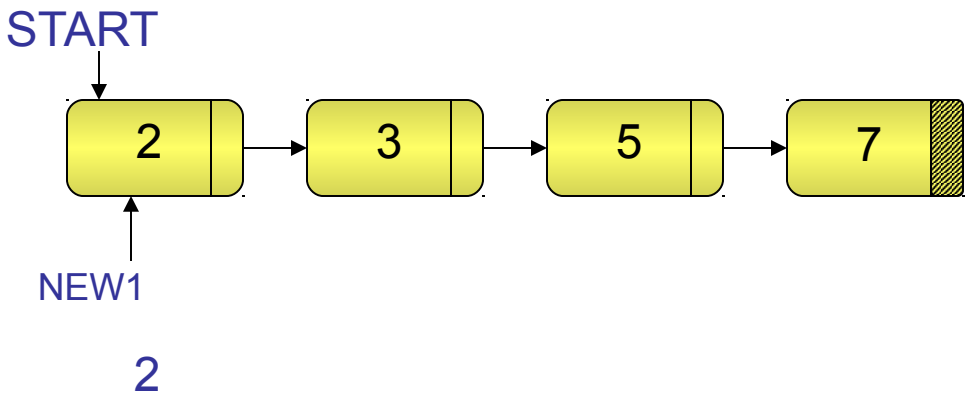
Traversing a Singly-Linked List (Contd.)

- Refer to the algorithm to display the elements in the linked list.



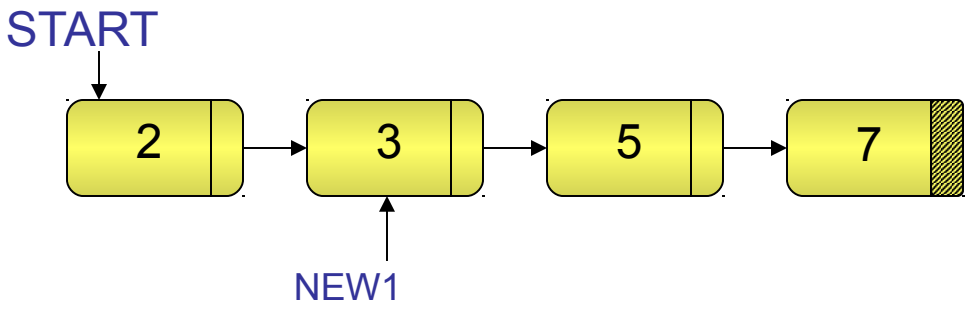
1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

Traversing a Singly-Linked List (Contd.)



1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

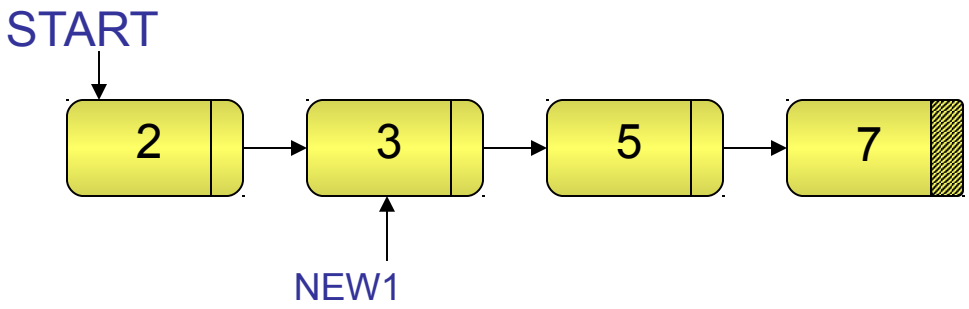
Traversing a Singly-Linked List (Contd.)



2

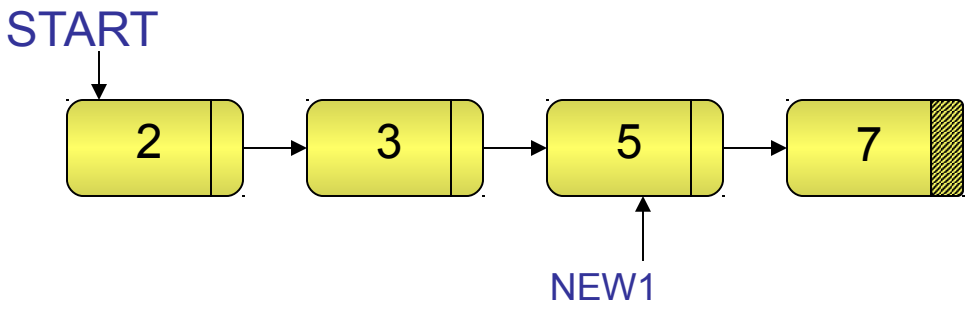
1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

Traversing a Singly-Linked List (Contd.)



1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

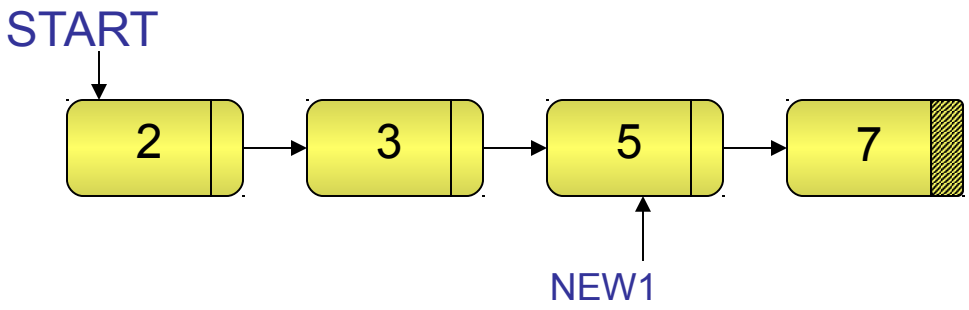
Traversing a Singly-Linked List (Contd.)



2 3

1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

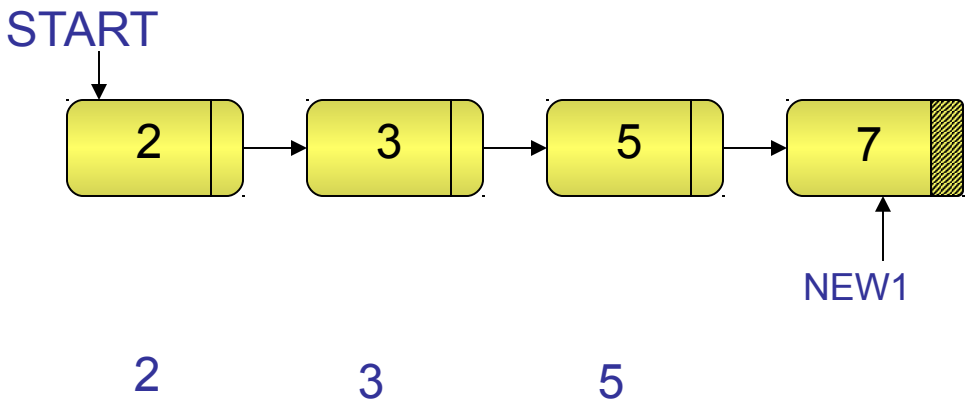
Traversing a Singly-Linked List (Contd.)



2 3 5

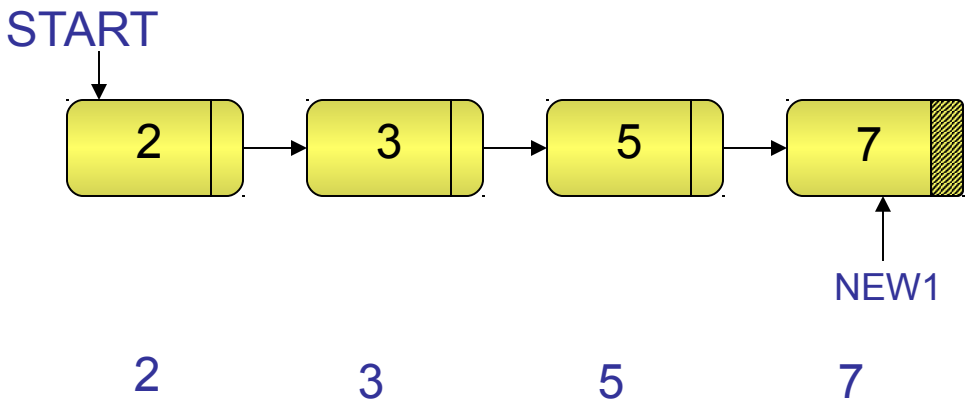
1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

Traversing a Singly-Linked List (Contd.)



1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

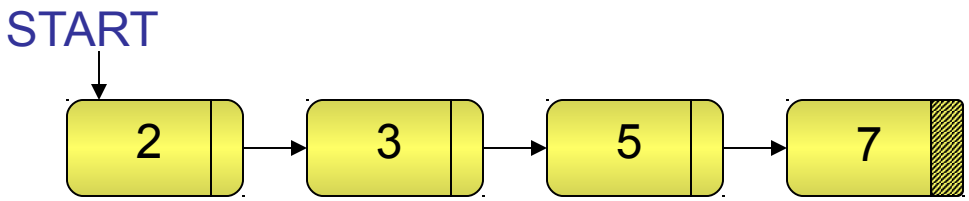
Traversing a Singly-Linked List (Contd.)



1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.

Traversing a Singly-Linked List (Contd.)

1. Make NEW1 point to the first node in the list.
2. Repeat step 3 and 4 until NEW1 becomes NULL.
3. Display the information contained in the node marked as NEW1
4. Make NEW1 point to the next node in sequence.



NEW1 = NULL

2 3 5 7

Traversal complete

TRAVERSING IN LIST

Algorithm traversing()

{

1.new1 = Start

2.while(new1 != NULL)

    2.1 Print new1 -> info

    2.2 new1 = new1 -> next

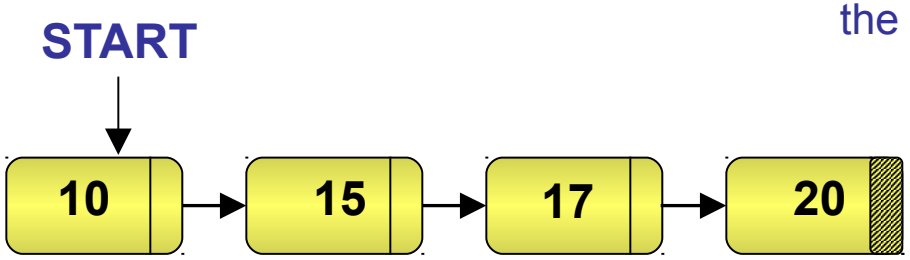
}

- Write an algorithm to insert a node in the beginning of a linked list.

Inserting a Node in a Singly-Linked List (Contd.)

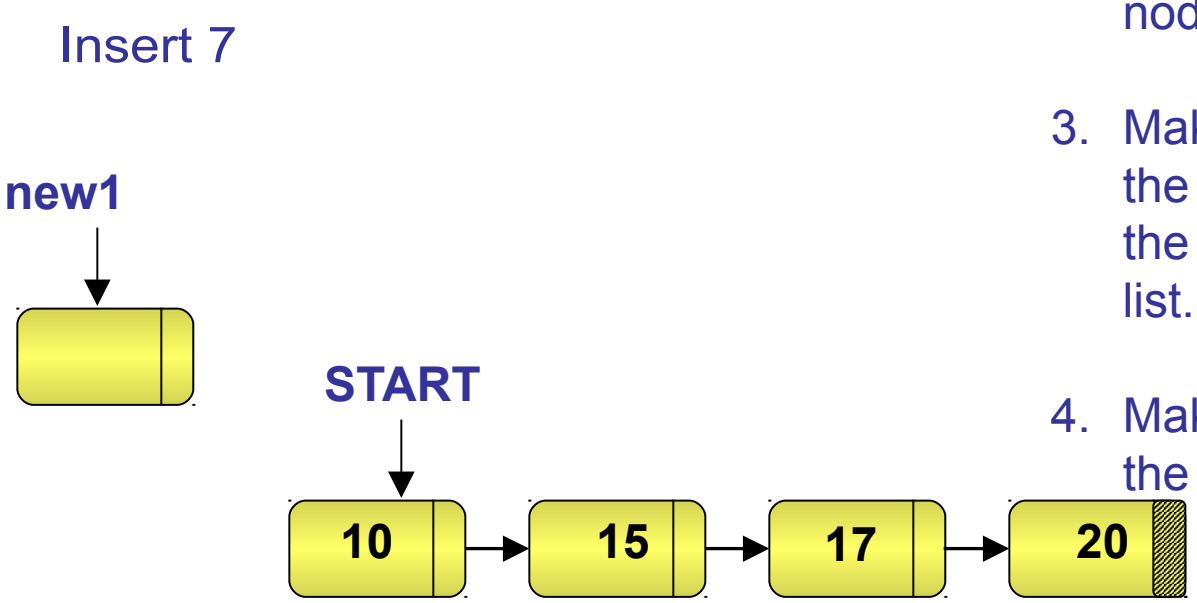
- Algorithm to insert a node in the beginning of a linked list

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to the first node in the list.
4. Make START, point to the new node.

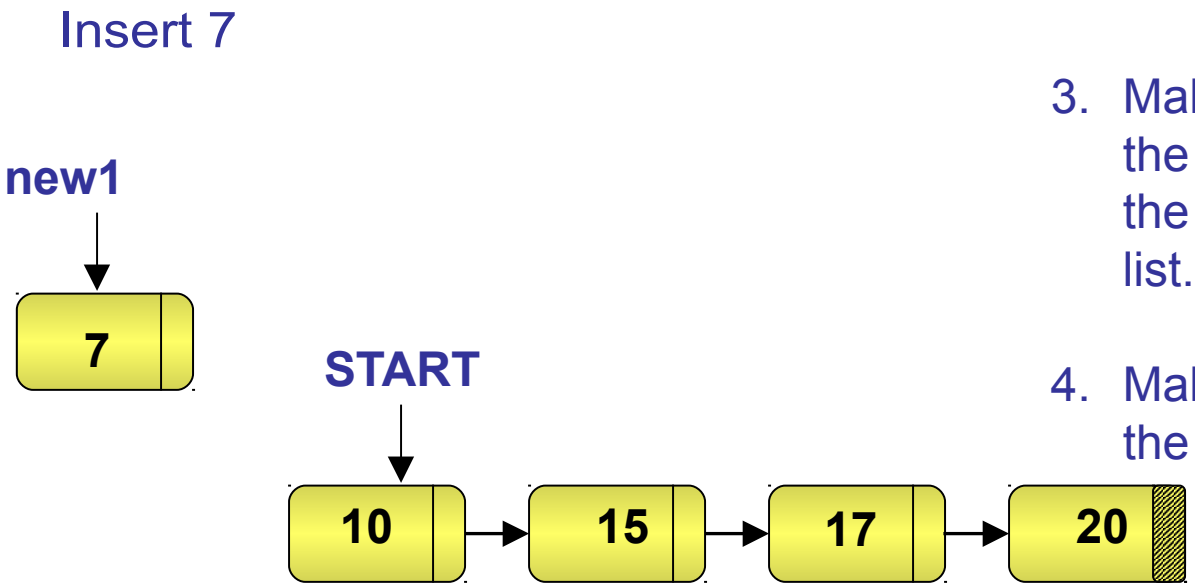


Inserting a Node in a Singly-Linked List (Contd.)

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to the first node in the list.
4. Make START, point to the new node.



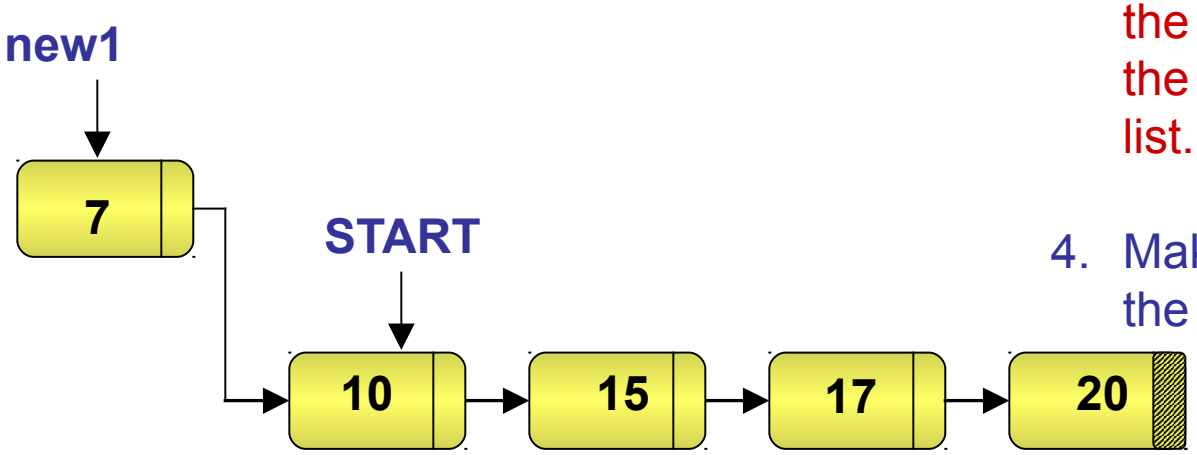
Inserting a Node in a Singly-Linked List (Contd.)



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to the first node in the list.
4. Make START, point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)

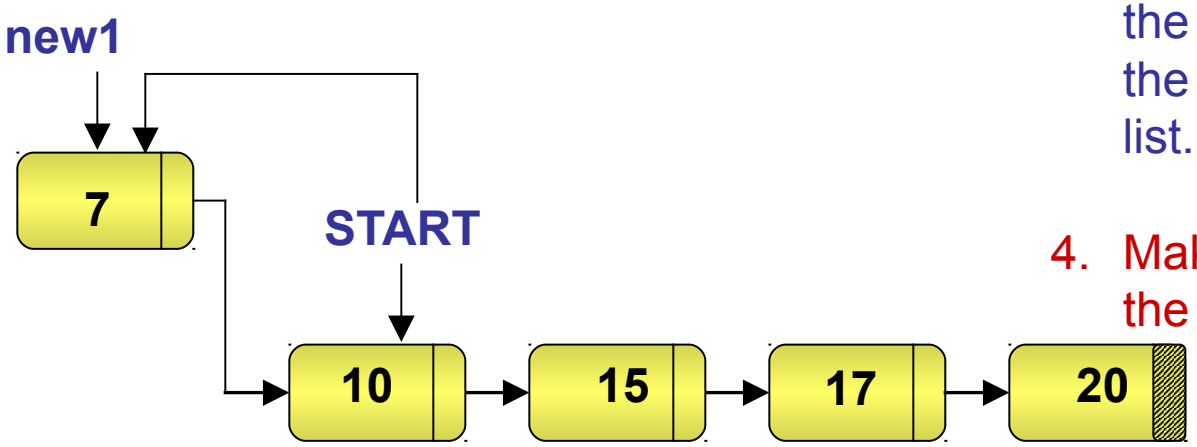
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to the first node in the list.
4. Make START, point to the new node.



new1 -> next = START

Inserting a Node in a Singly-Linked List (Contd.)

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to the first node in the list.
4. Make START, point to the new node.



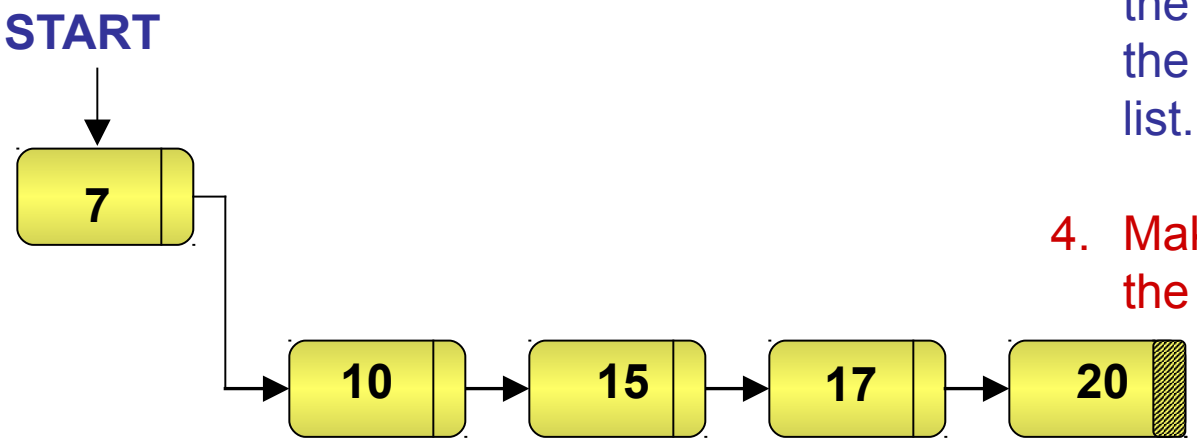
Insertion complete

New1 -> next = START

START = new1

Inserting a Node in a Singly-Linked List (Contd.)

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to the first node in the list.
4. Make START, point to the new node.



Insertion complete

New1 -> next = START

START = new1

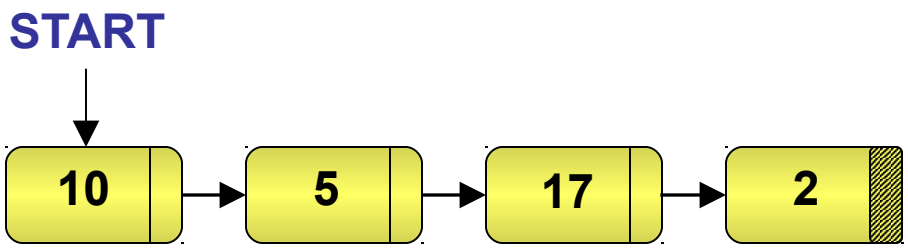
ALGORITHM TO INSERT NODE AT BEGINING

```
Start=NULL
Algorithm InsertAtBEG()
{
1. Create node [(new1=(struct node*) malloc(sizeof(struct node))]
2. Enter data [new1 -> info = data]
3. If (Start == NULL)
            3.1 new1 -> next = NULL
            3.2 Last=new1
            3.3 Start=new1
   else
            3.1 new1 -> next=Start
            3.2 Start = new1
}
```

- Write an algorithm to insert a node at the particular position in a linked list.

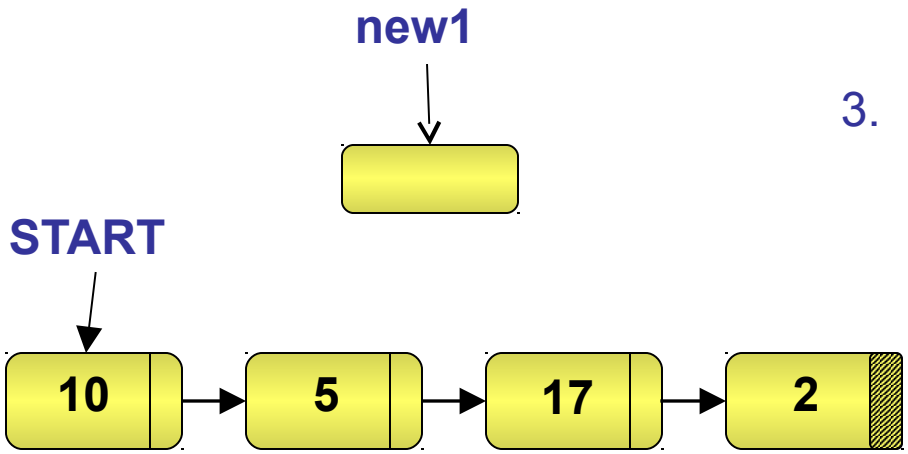
Inserting a Node in a Singly-Linked List (Contd.)

Insert 16 At LOC = 3



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence
4. Make the next field of the new node point to the next of previous node
5. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



1.Allocate memory for the new node.

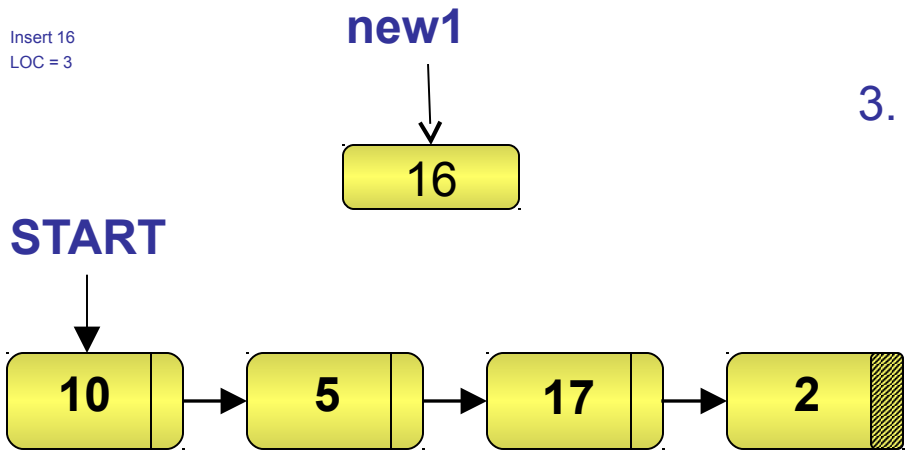
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



1.Allocate memory for the new node.

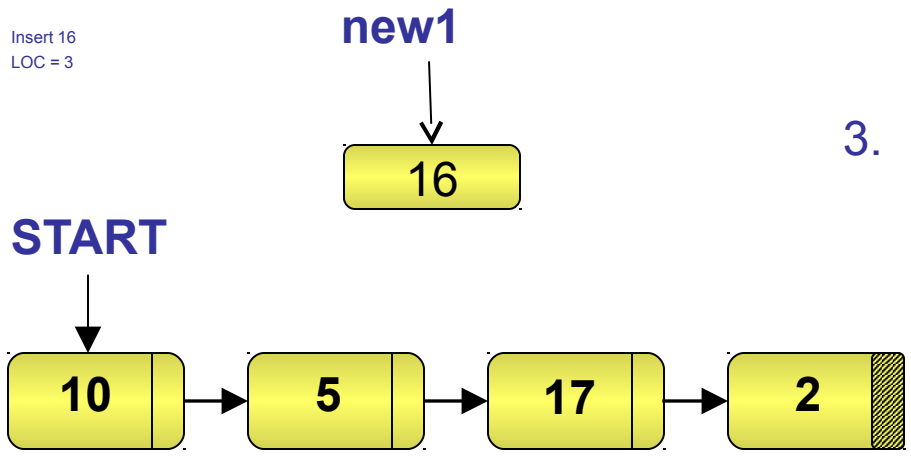
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



1.Allocate memory for the new node.

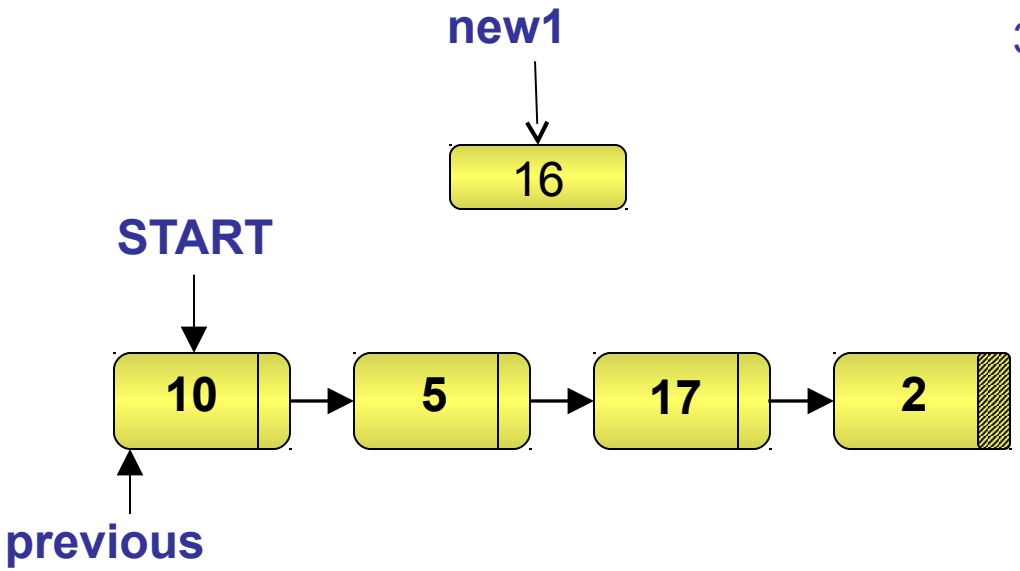
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



Count=1

1.Allocate memory for the new node.

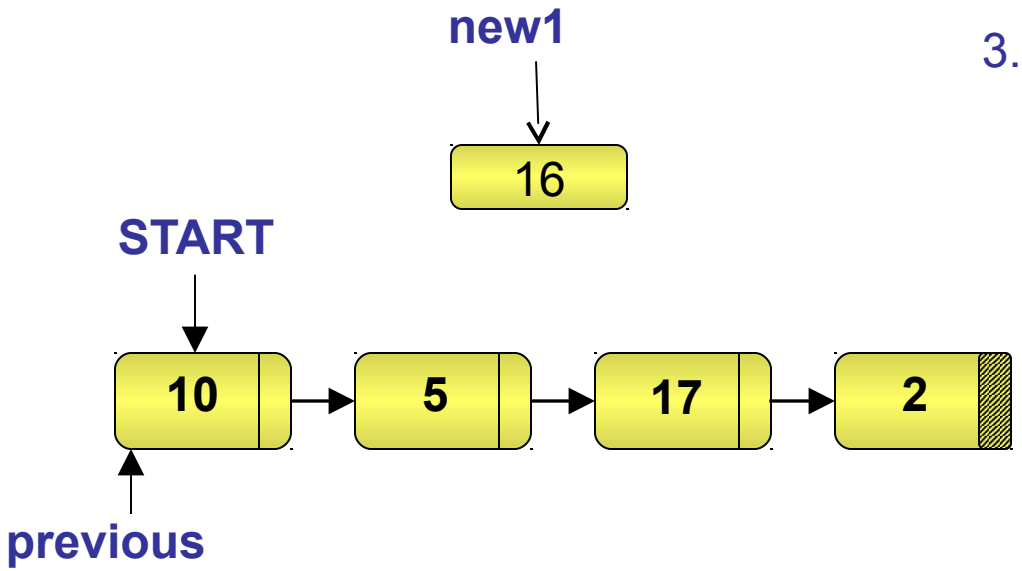
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



Count=1

1.Allocate memory for the new node.

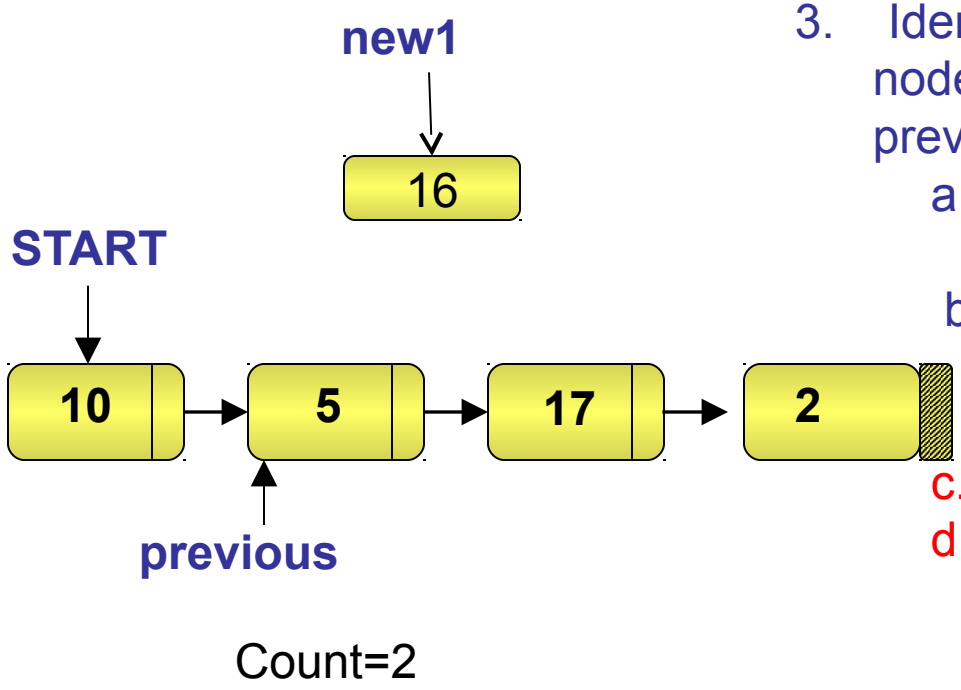
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



1.Allocate memory for the new node.

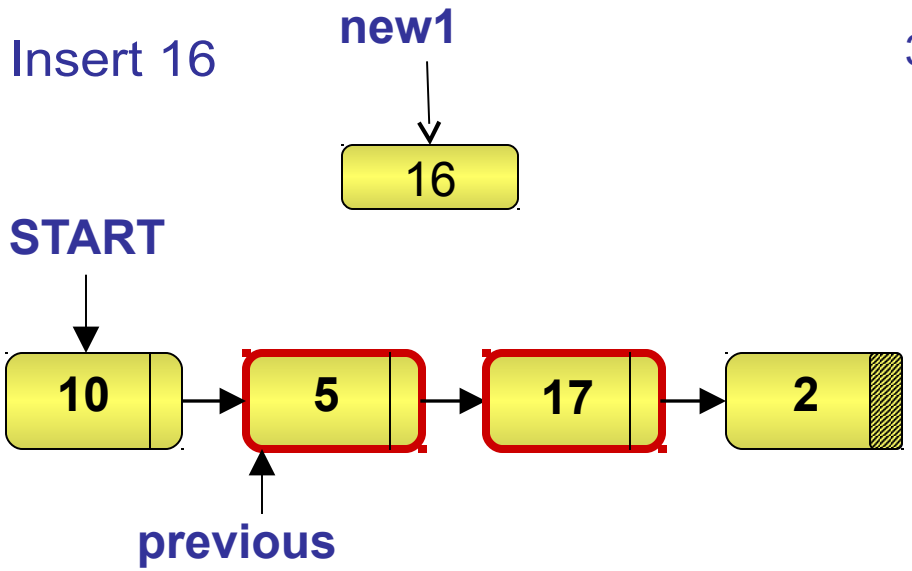
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

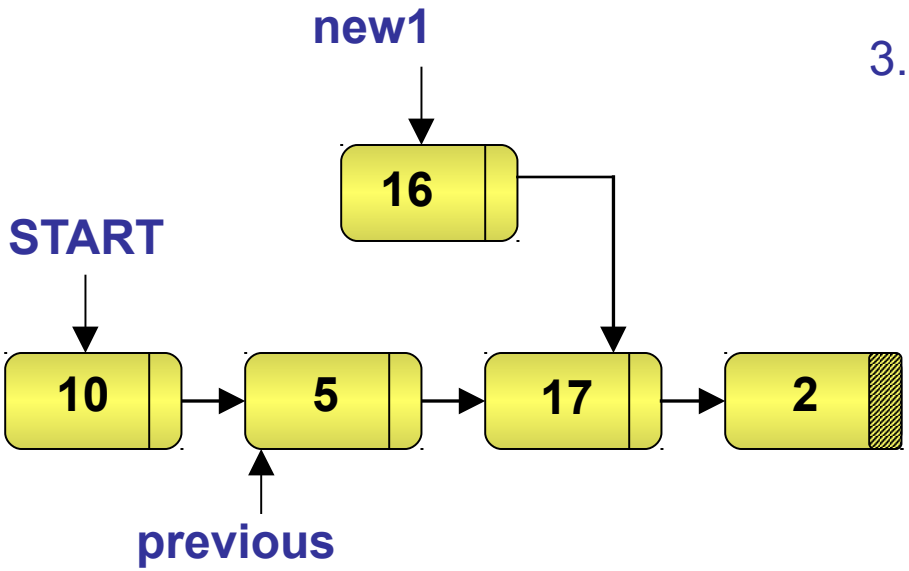
Inserting a Node in a Singly-Linked List (Contd.)



Nodes located

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence
4. Make the next field of the new node point to the next of previous node
5. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



new1 -> next = previous->next

1.Allocate memory for the new node.

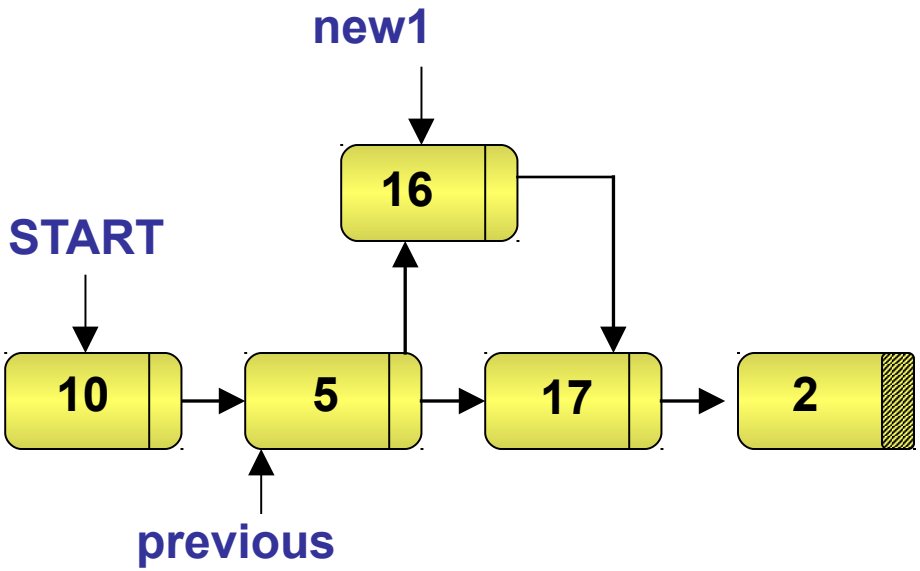
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



new1 -> next = previous->next

previous -> next = new1

1.Allocate memory for the new node.

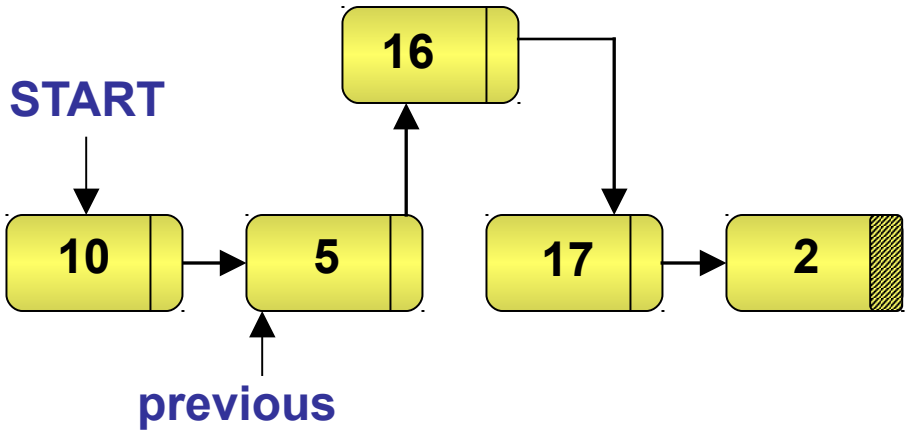
2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5.Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



new1 -> next = previous->next

previous -> next = new1

Insertion complete

1. Allocate memory for the new node.

2. Assign value to the data field of the new node.

3. Identify the nodes after which the new node is to be inserted. Mark it as previous
   a. Make previous node point to the first node and set count=1
   b. Repeat step c and step d until count becomes equal to location-1
   c. Count=count+1.
   d. Make previous point to next node in sequence

4. Make the next field of the new node point to the next of previous node

5. Make the next field of previous point to the new node.

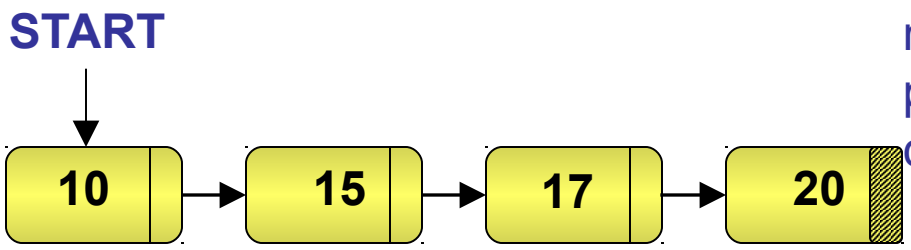
ALGORITHM TO INSERT NODE At PARTICULAR POSITION IN A LINKED LIST

```
Algorithm InsertAtSpec()
{
1. Create node [(new1=(struct node*) malloc(sizeof(struct node))]
2. Enter Data and Location
3. new1->info=Data
4. If (Location == 1)
        4.1 new1 -> next = Start
        4.2 Start = new1
    Else
        4.1 Previous = Start
        4.2 Count = 1
        4.3 While( Count <= Location - 1 && Previous->next !=NULL)
                    4.3.1 Previous = Previous -> next
                    4.3.2 Count++
        4.4 new1 -> next = Previous -> next
        4.5 Previous -> next = new1
}
```

- Write an algorithm to insert a node in a Sorted linked list.

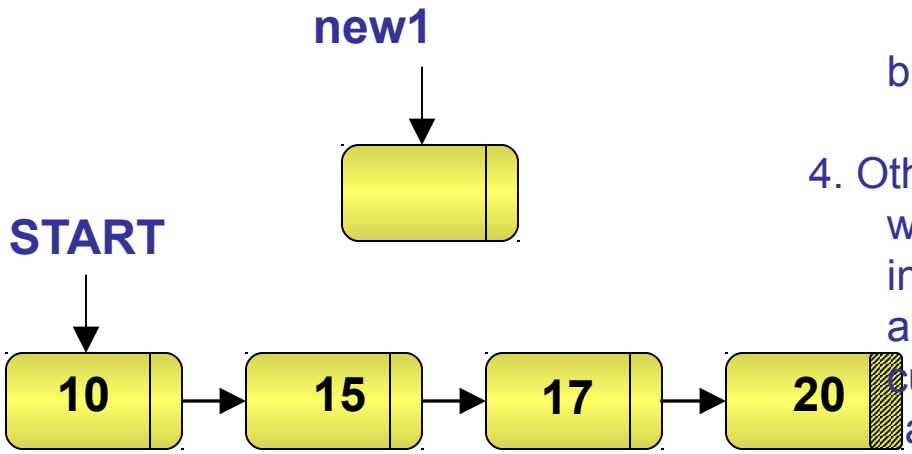
Inserting a Node in a Singly-Linked List (Contd.)

Insert 16



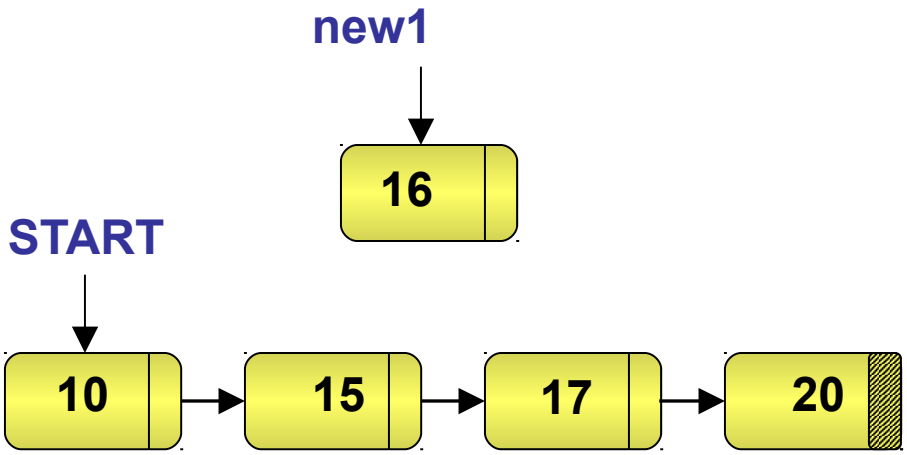
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



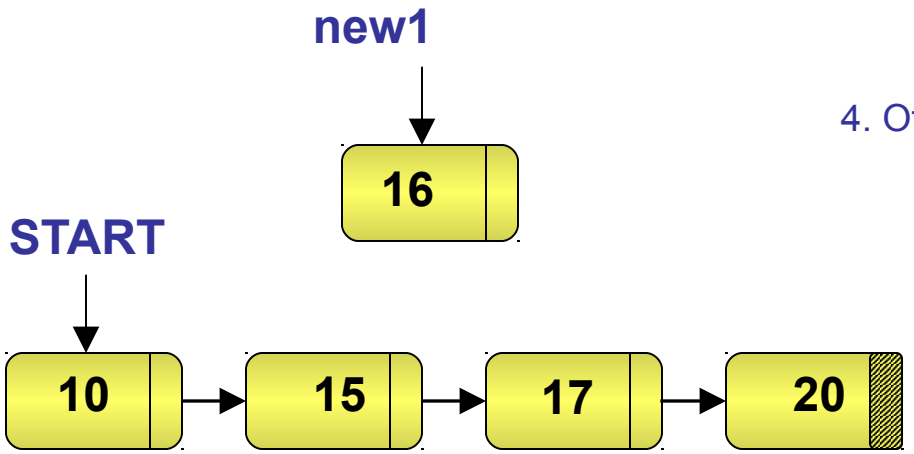
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

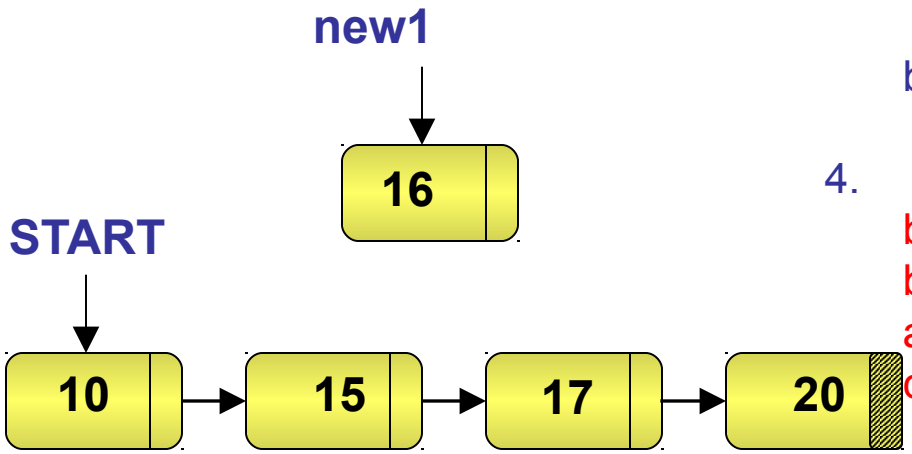
Inserting a Node in a Singly-Linked List (Contd.)



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the

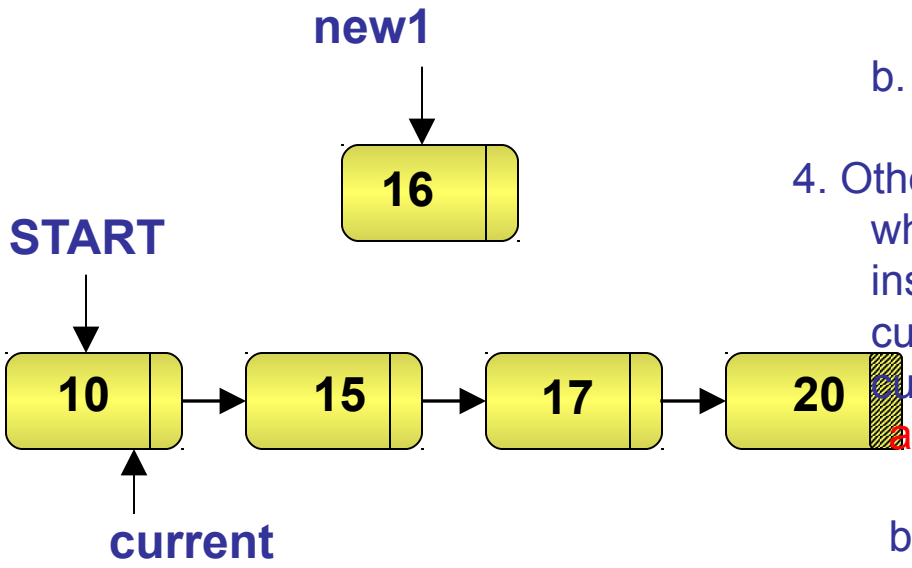
new node.

Inserting a Node in a Singly-Linked List (Contd.)



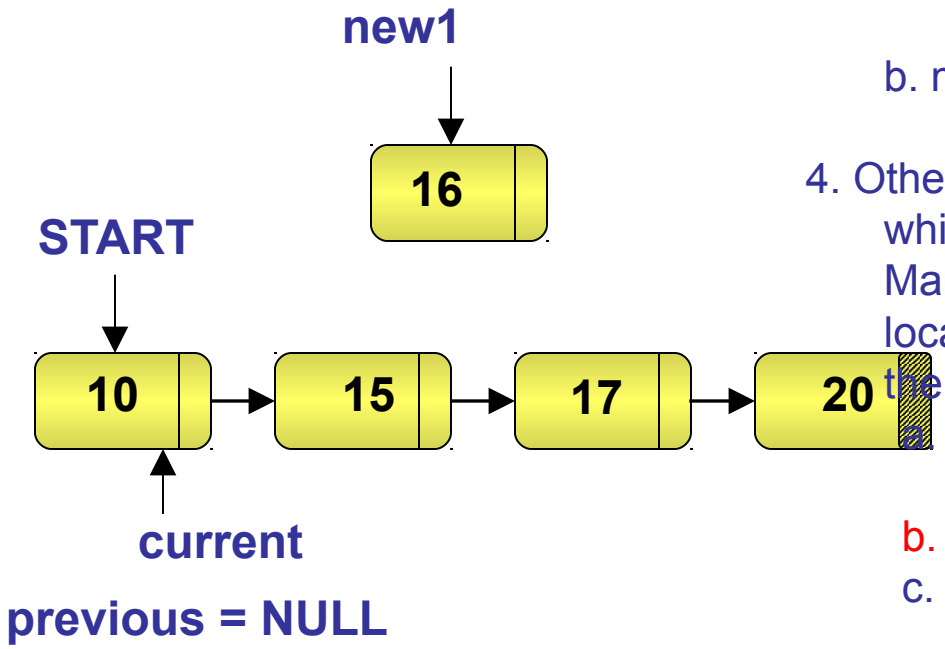
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



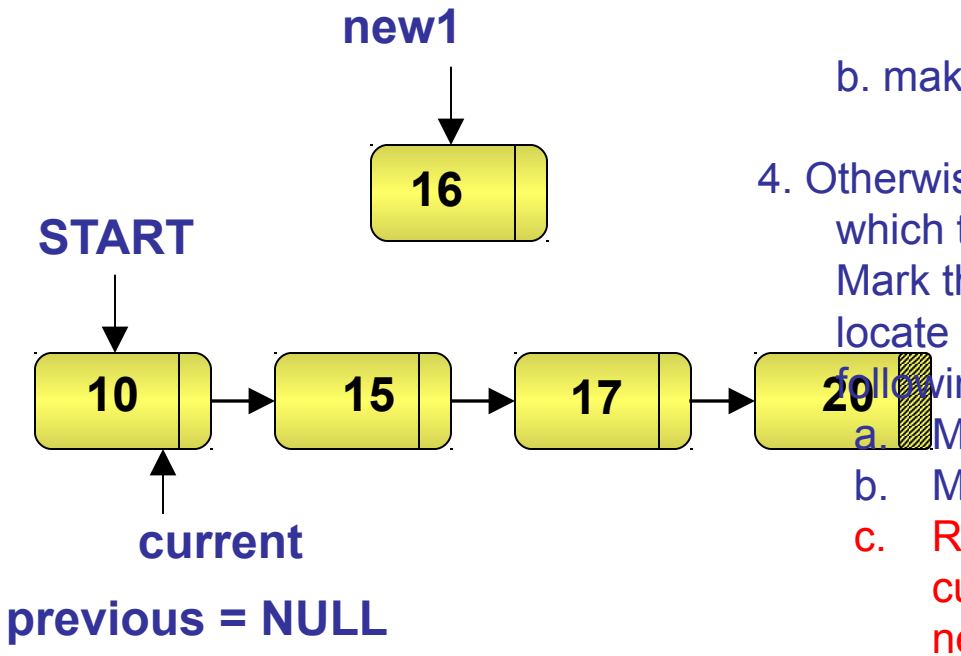
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



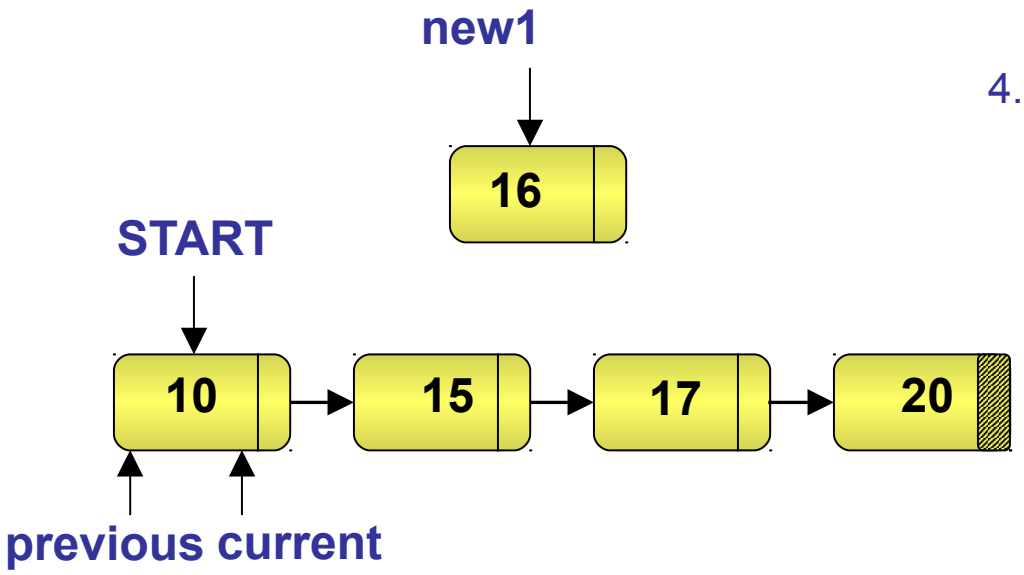
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



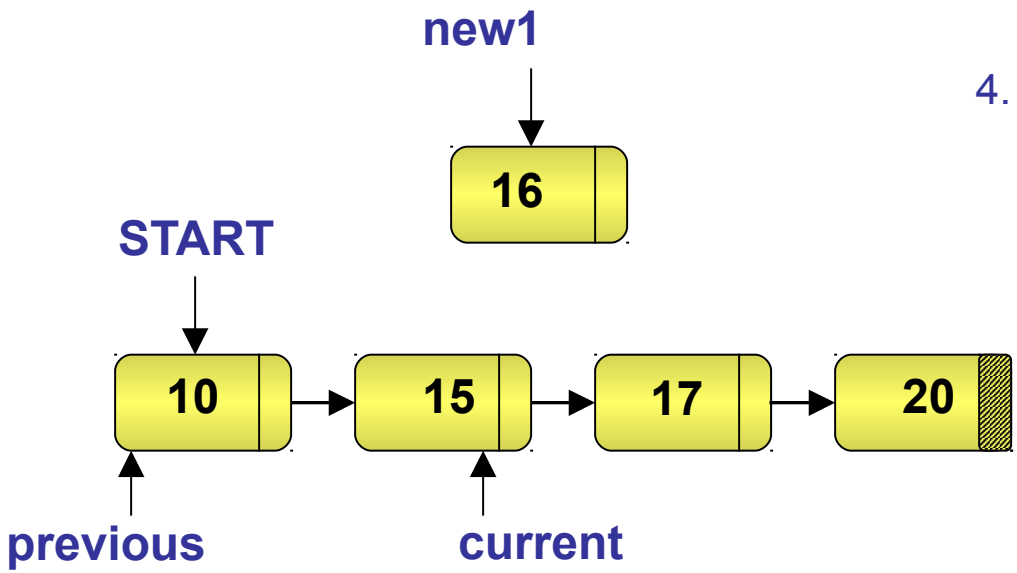
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



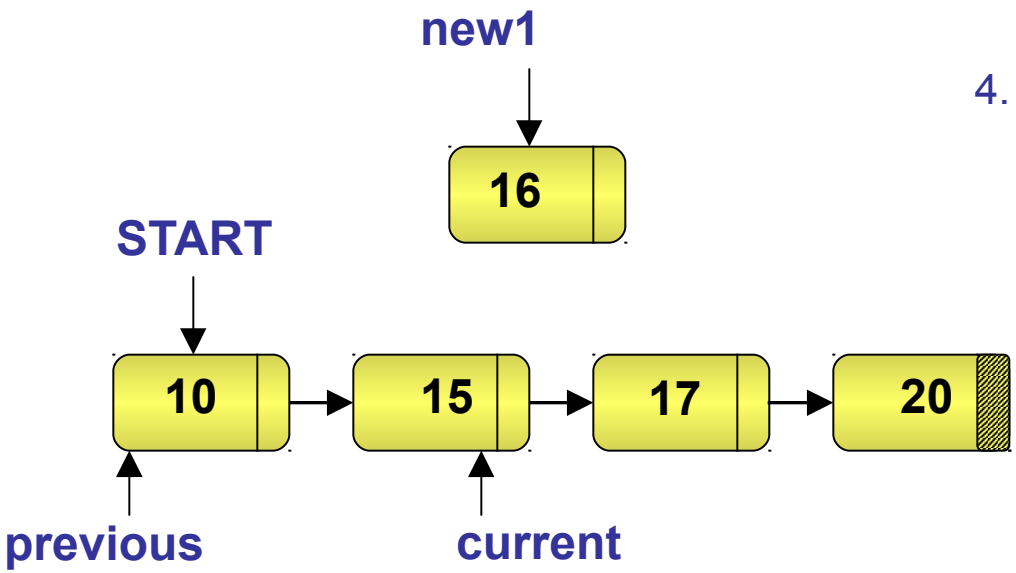
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



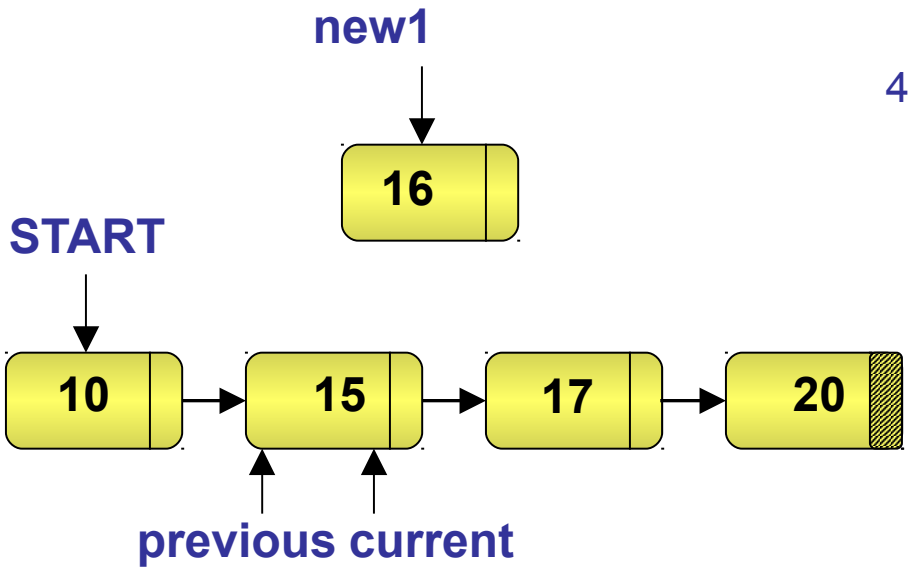
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



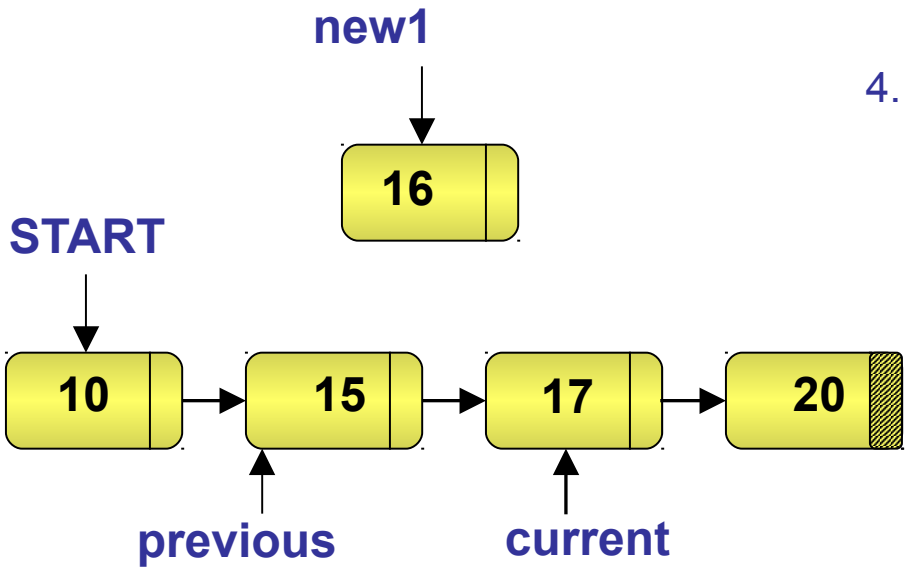
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



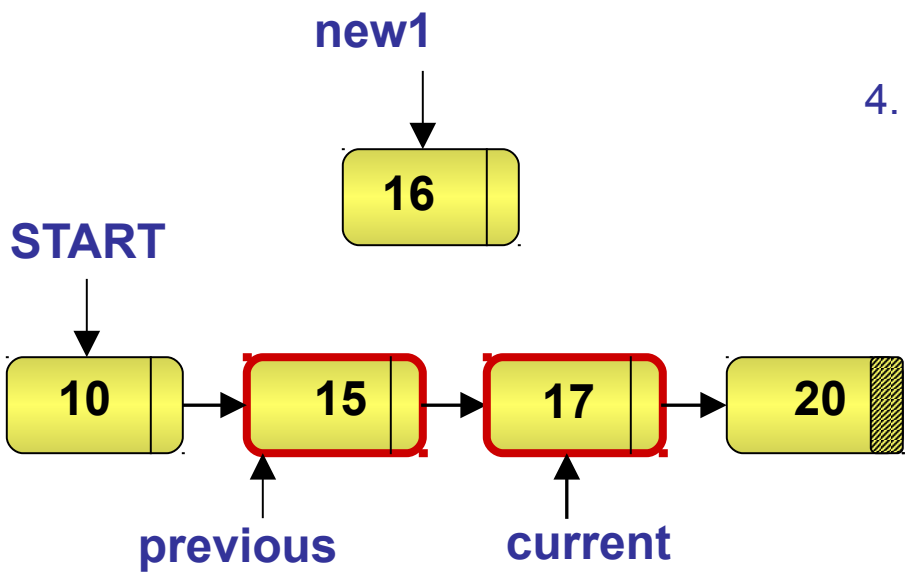
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

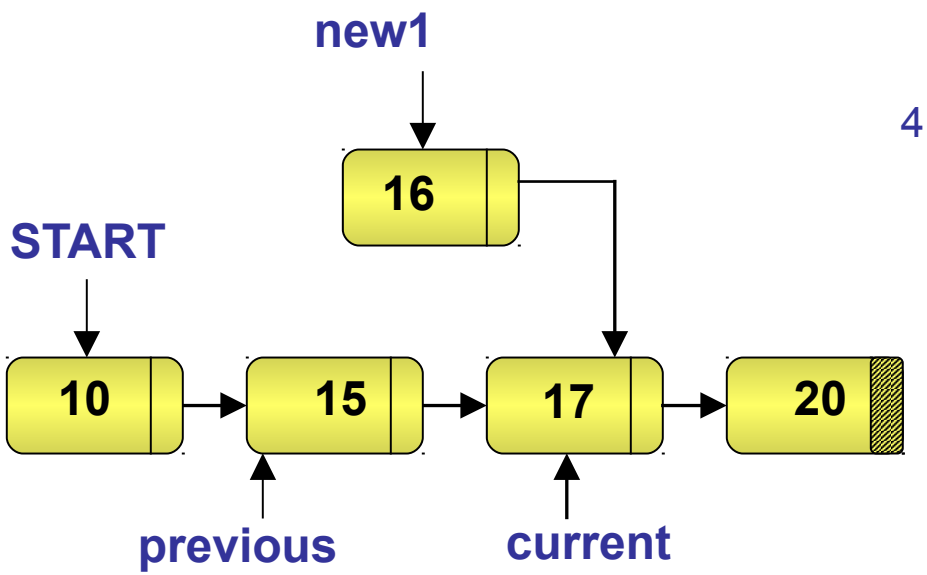
Inserting a Node in a Singly-Linked List (Contd.)



Nodes located

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

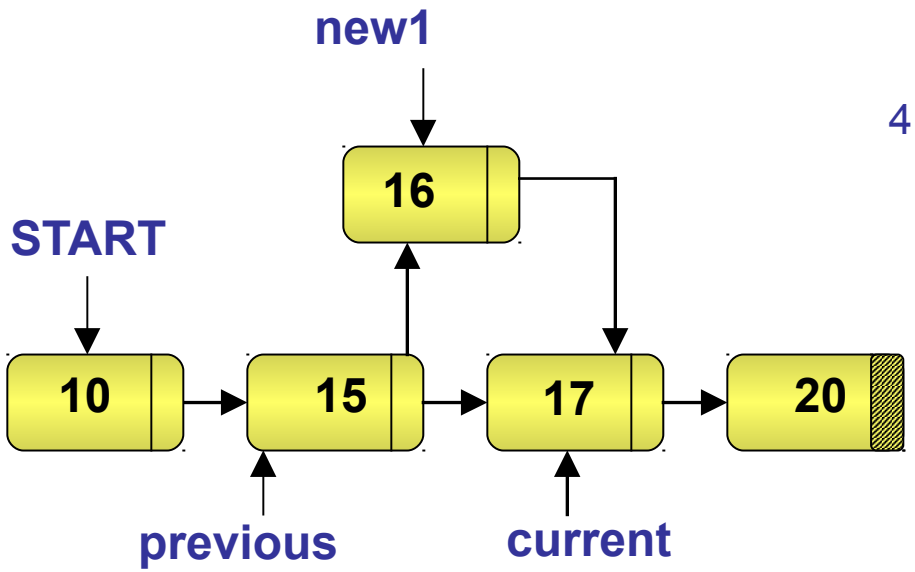
Inserting a Node in a Singly-Linked List (Contd.)



new1 -> next = current

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



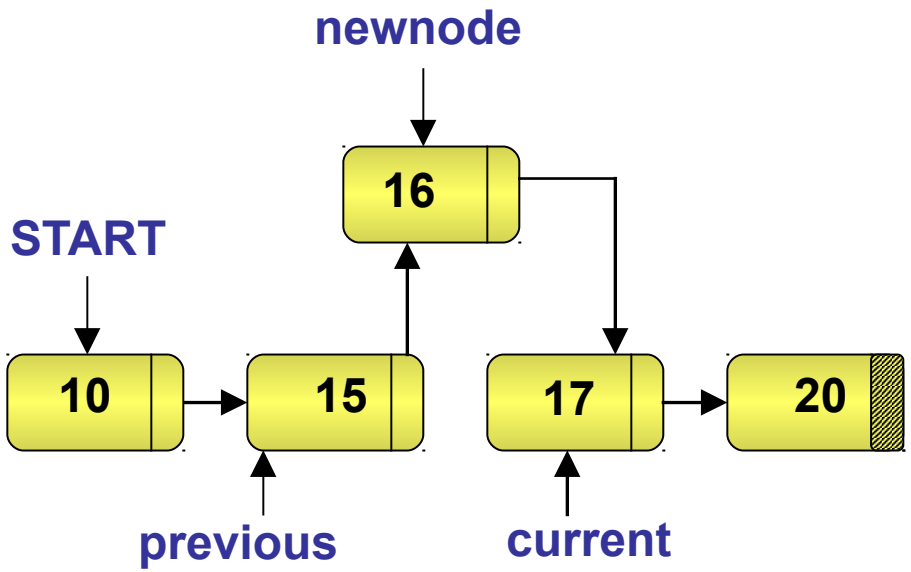
new1 -> next = current

previous -> next = new1

Insertion complete

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

Inserting a Node in a Singly-Linked List (Contd.)



new1 -> next = current

previous -> next = new1

Insertion complete

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. If data of newnode is less than the data of start node, Then
   a. make newnode point to the start node
   b. make start point to the newnode
4. Otherwise,Identify the nodes between which the new node is to be inserted. Mark them as previous and current. To locate previous and current, execute the following steps:
   a. Make current point to the first node.
   b. Make previous point to NULL.
   c. Repeat step d and step e until current.info becomes greater than newnode.info or current becomes equal to NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
5. Make the next field of the new node point to current.
6. Make the next field of previous point to the new node.

ALGORITHM TO INSERT NODE IN A SORTED LINKED LIST

```
Algorithm InsertAtSorted()
{
1. Enter Data
2. Create node [(new1=(struct node*) malloc(sizeof(struct node))]
3. new1->info=Data
4. If (Data <= Start -> info)
        4.1 new1 -> next = Start
        4.2 Start = new1
    Else
        4.1 Current = Start
        4.2 Previous = NULL
        4.3While( Data >= Current -> info && Current  != NULL)
                    4.3.1 Previous = Current
                    4.3.2 Current = Current -> next

        4.4 new1 -> next = Current
        4.5 Previous -> next = new1
}
```

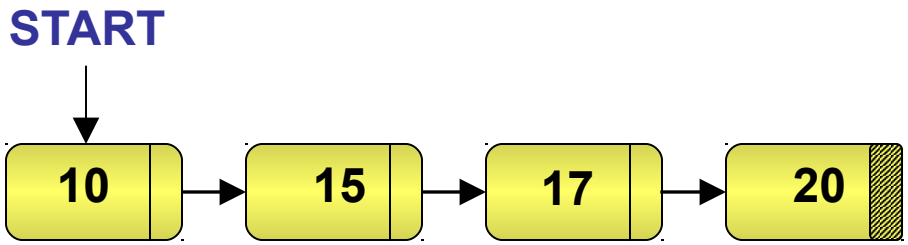
Deleting a Node from a Singly-Linked List

- Delete operation in a linked list refers to the process of removing a specified node from the list.
- You can delete a node from the following places in a linked list:
  - Beginning of the list
  - Between two nodes in the list
  - End of the list

- Write an algorithm to delete the first node in a linked list.

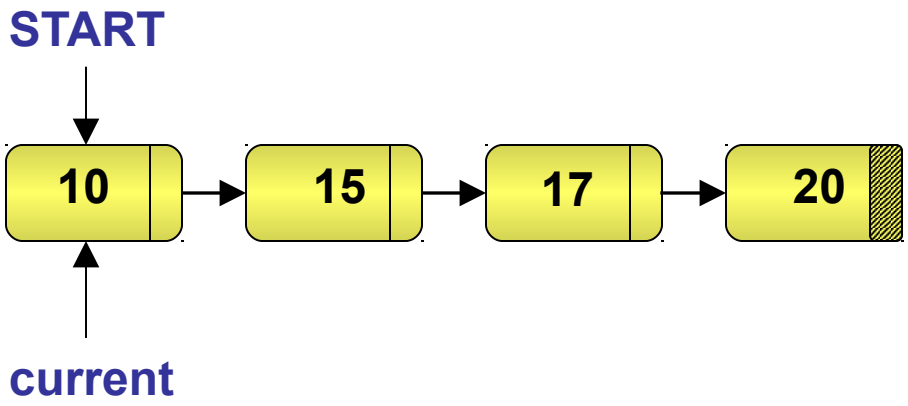
Deleting a Node From the Beginning of the List (Contd.)

- Algorithm to delete a node from the beginning of a linked list.



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

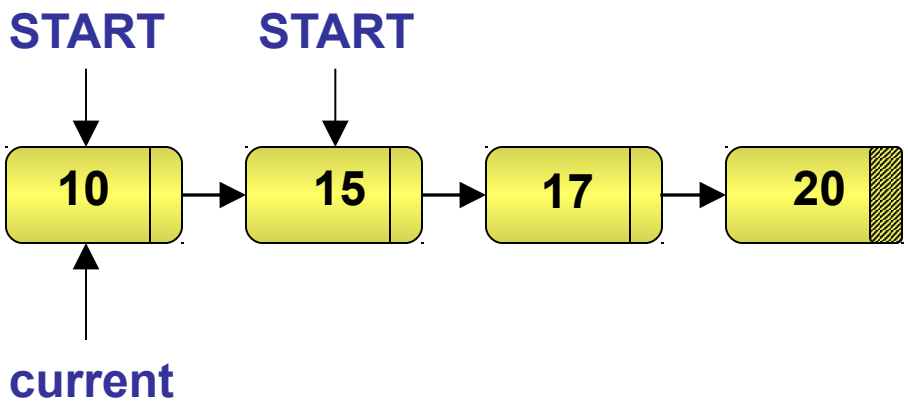
Deleting a Node From the Beginning of the List (Contd.)



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

current = START

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



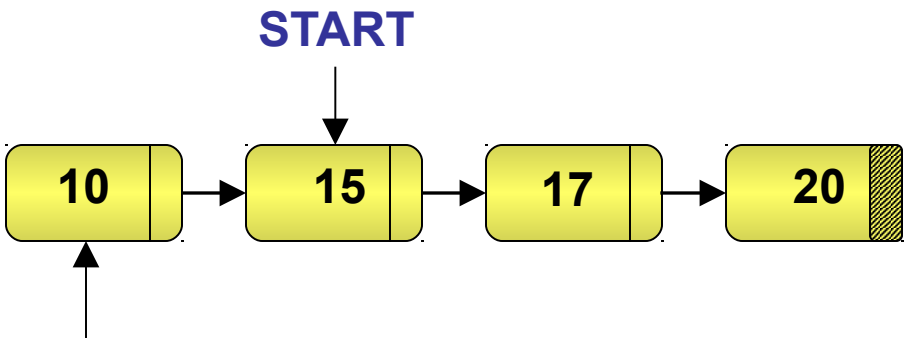
1. Mark the first node in the list as current.
2. Make START point to the next node in its sequence.
3. Release the memory for the node marked as current.

current = START

START = START -> next

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

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



Memory released

current = START

START = START -> next

Delete operation complete

ALGORITHM TO DELETE A NODE FROM THE BEGINNING

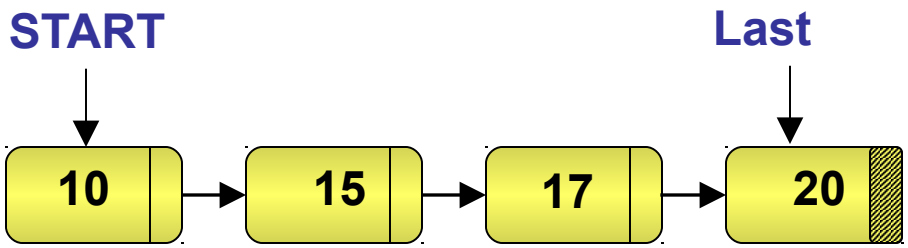
```
Algorithm DeleteAtBeg()
{
1. If (Start == NULL)
        1.1 Print "underflow“
   else
        1.1 Current = Start
        1.2 Start = Start -> next
        1.3 Current -> next = NULL
        1.4 Release the memory [ free (Current) ]

}
```

- Write an algorithm to delete the End node in a linked list.

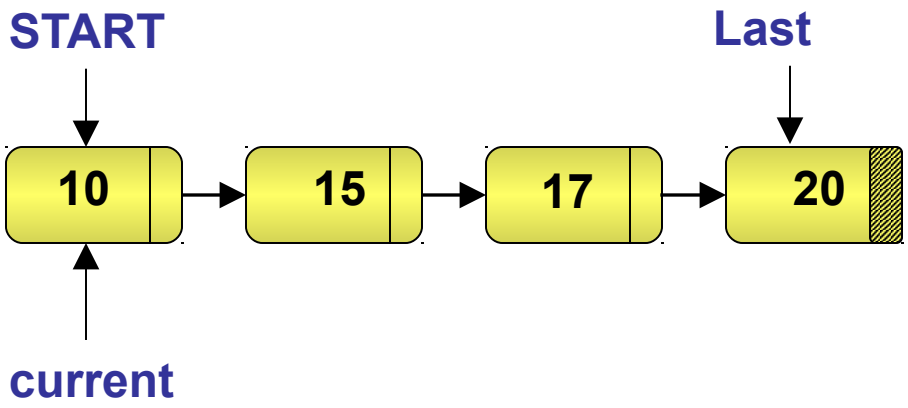
Deleting a Node From the Beginning of the List (Contd.)

- Algorithm to delete a node from the End.



1. Mark the first node in the list as current..
2. Repeat step 3 untill current becomes Last node.
3. Make current point to the next node in its sequence.
4. Release the memory for the node marked as Last.
5. Make Current point to the last node

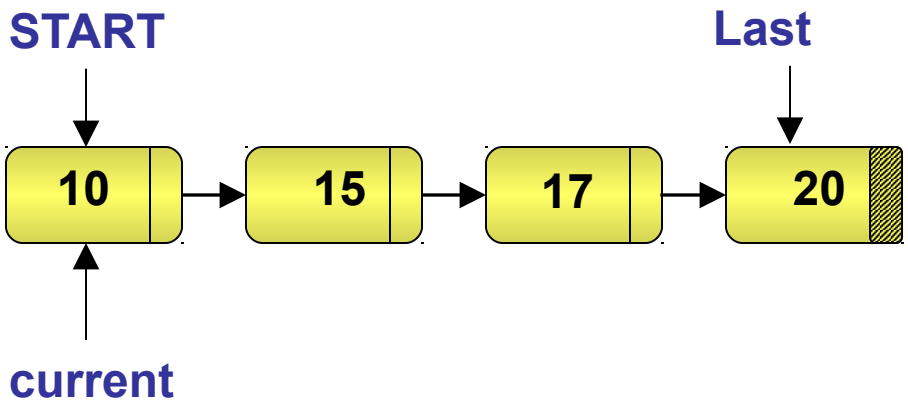
Deleting a Node From the Beginning of a Linked List (Contd.)



1. Mark the first node in the list as current..
2. Repeat step 3 untill current becomes Last node.
3. Make current point to the next node in its sequence.
4. Release the memory for the node marked as Last.
5. Make Current point to the last node

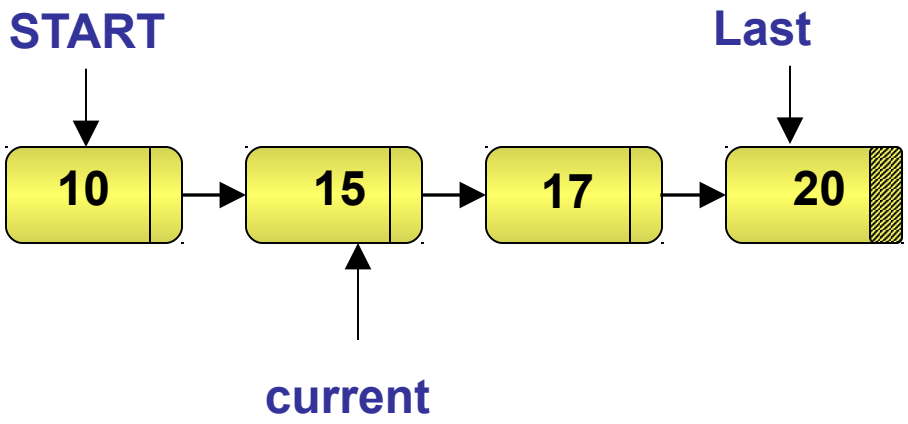
current = START

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



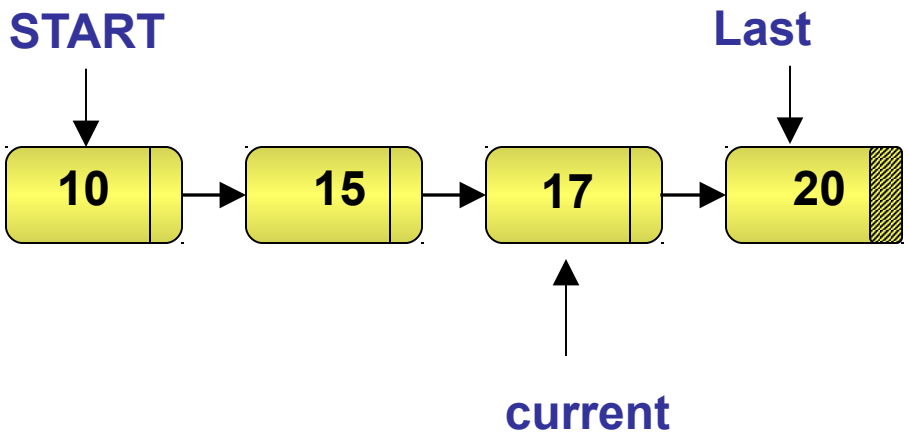
1. Mark the first node in the list as current..
2. Repeat step 3 untill current becomes Last node.
3. Make current point to the next node in its sequence.
4. Release the memory for the node marked as Last.
5. Make Current point to the last node

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



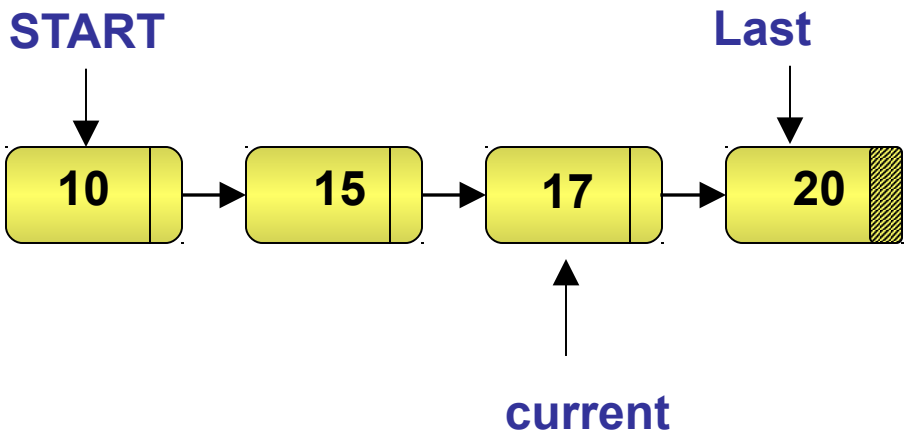
1. Mark the first node in the list as current..
2. Repeat step 3 untill next of current becomes Last node.
3. Make current point to the next node in its sequence.
4. Release the memory for the node marked as Last.
5. Make Current point to the last node

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



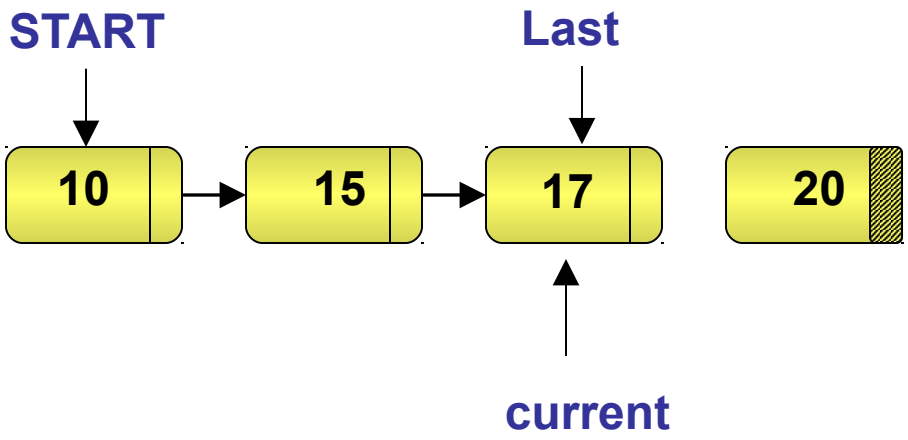
1. Mark the first node in the list as current..
2. Repeat step 3 untill next of current becomes Last node.
3. Make current point to the next node in its sequence.
4. Release the memory for the node marked as Last.
5. Make Current point to the last node

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



1. Mark the first node in the list as current..
2. Repeat step 3 untill next of current becomes Last node.
3. Make current point to the next node in its sequence.
4. Release the memory for the node marked as Last.
5. Make Current point to the last node

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



1. Mark the first node in the list as current..
2. Repeat step 3 untill next of current becomes Last node.
3. Make current point to the next node in its sequence.
4. Release the memory for the node marked as Last.
5. Make Current point to the last node

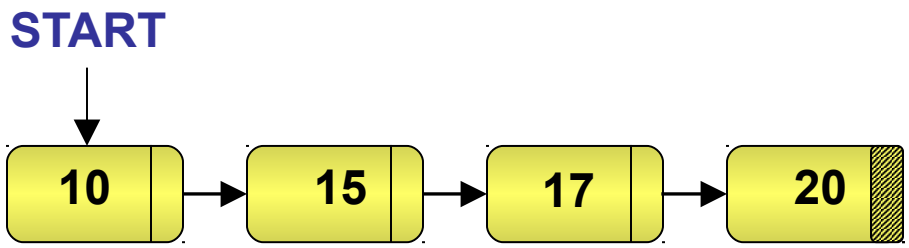
Delete operation complete

ALGORITHM TO DELETE A NODE FROM THE END

```
Algorithm DeleteAtEnd()
{
1. If (Start == NULL)
        1.1 Print "underflow“
   else If (Start -> next == NULL )
        1.1 Release the memory [ free (Start) ]
        1.2 Start == NULL
    else
        1.1 Current = Start
        1.2 while ( Current -> next != Last )
                1.2.1 Current = Current -> next
        1.3 Current -> next = NULL
        1.4 Release the memory [ free (Last) ]
        1.5 Last = Current
}
```

- Write an algorithm to delete a node from a specific position in a linked list.

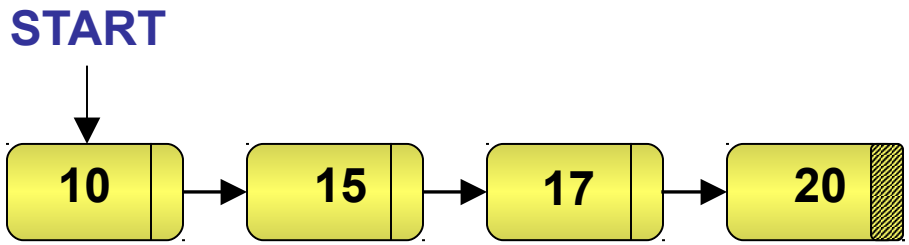
- Algorithm to delete a node from a specific position.
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
    a. Set previous = NULL
    b. Set current = START
    c. Repeat step d and e until either the node is found or current becomes NULL.
    d. Make previous point to current .
    e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

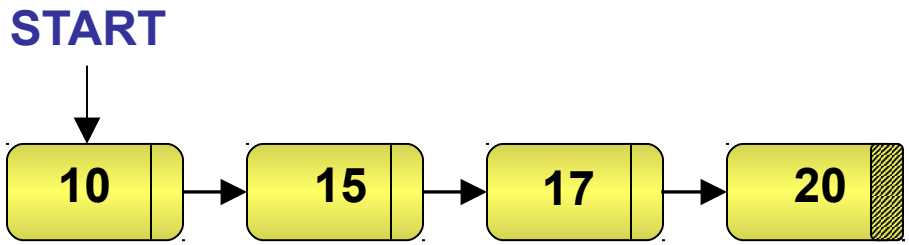
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

- Delete 17

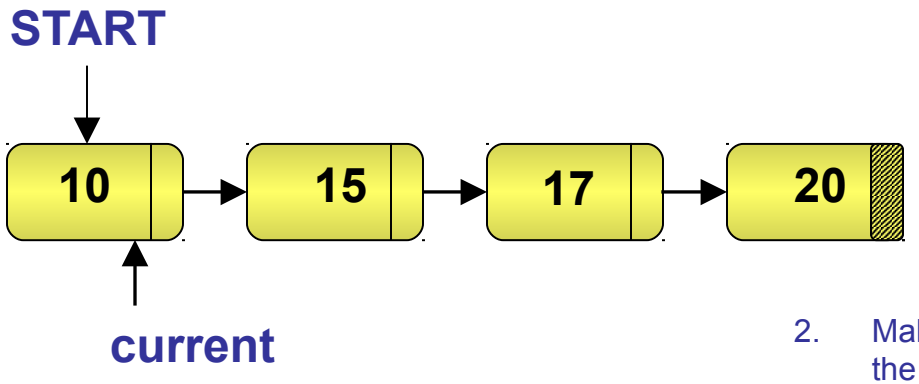


Previous = NULL

1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

- Delete 17

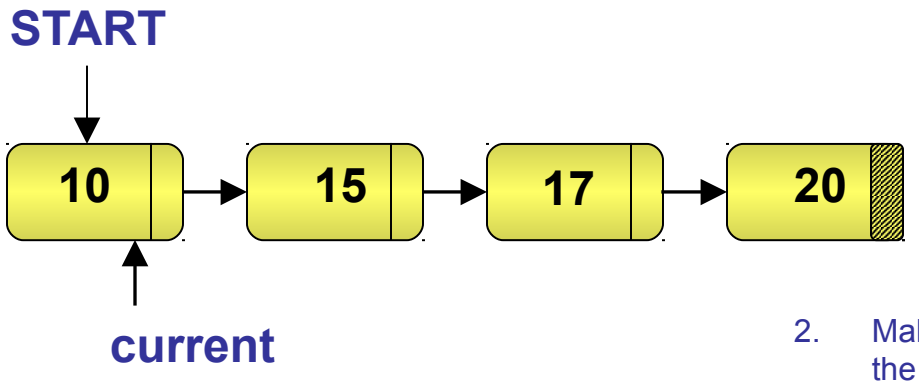


Previous = NULL

1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

- Delete 17

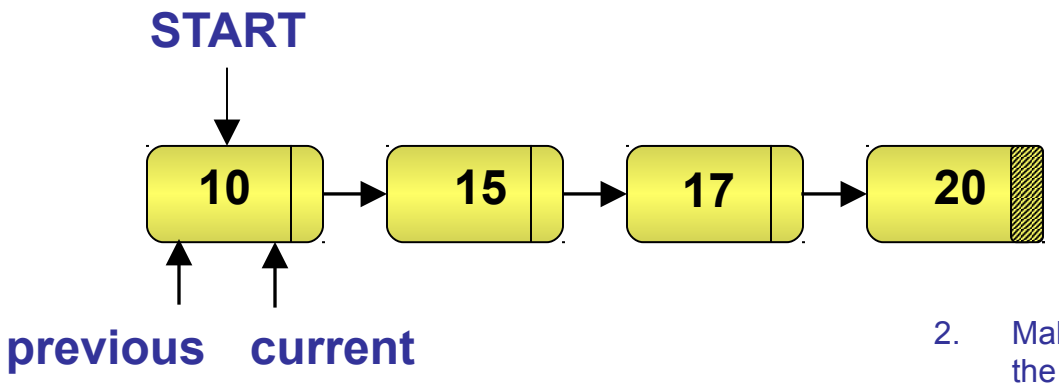


Previous = NULL

1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

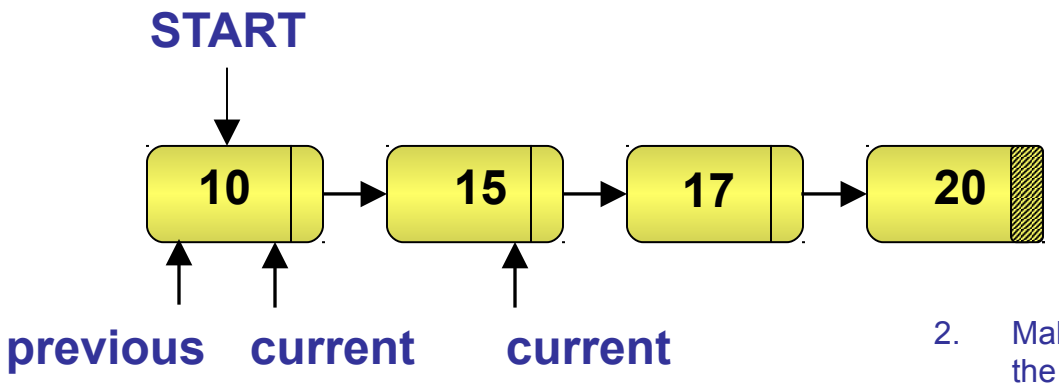
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

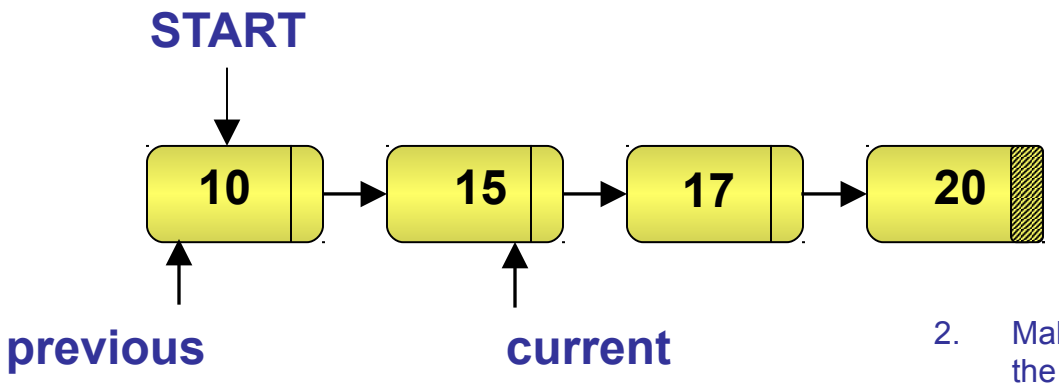
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

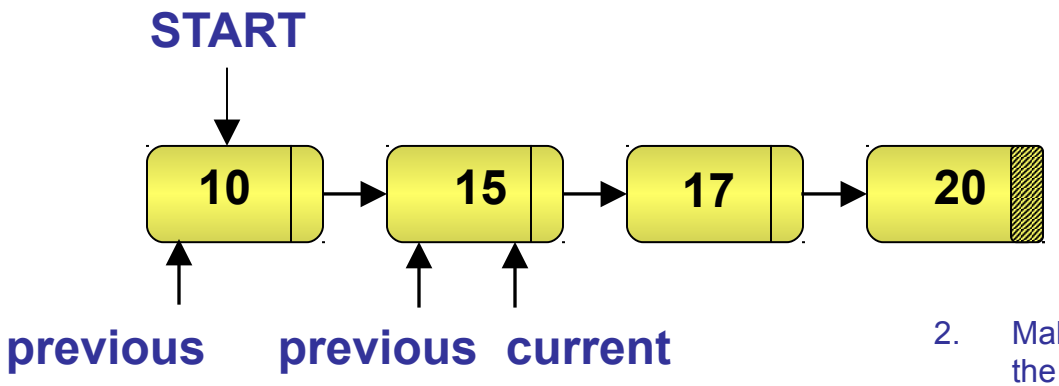
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

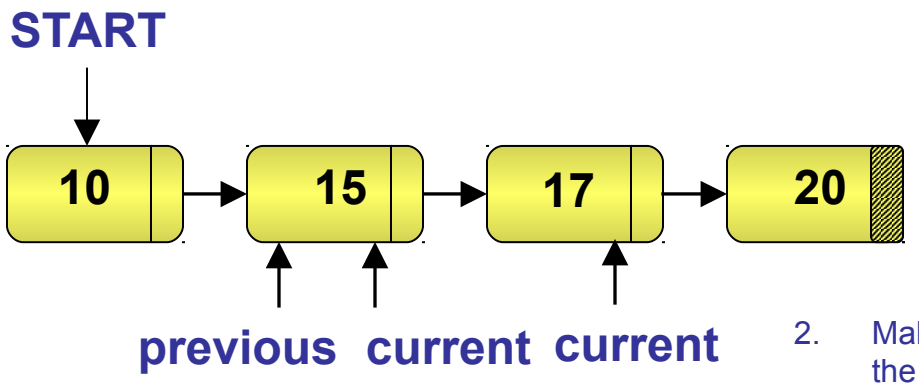
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

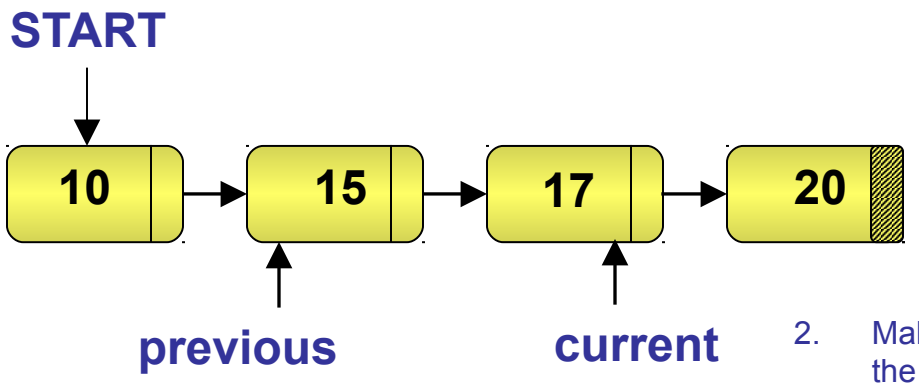
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

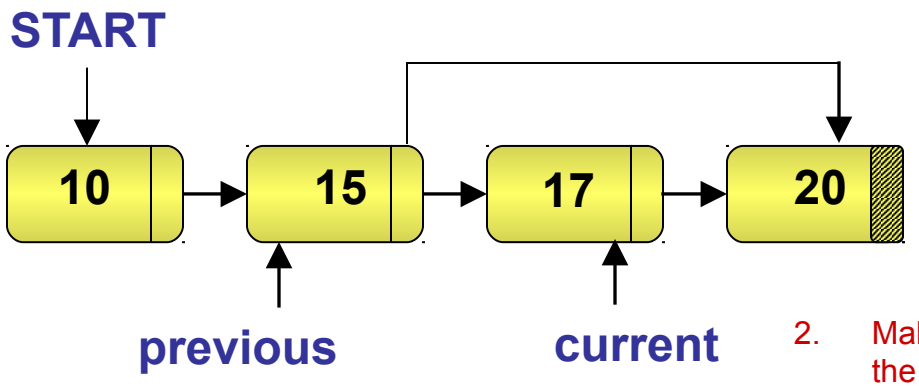
- Delete 17



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Deleting a Node Between two Nodes in the List (Contd.)

- Delete 17



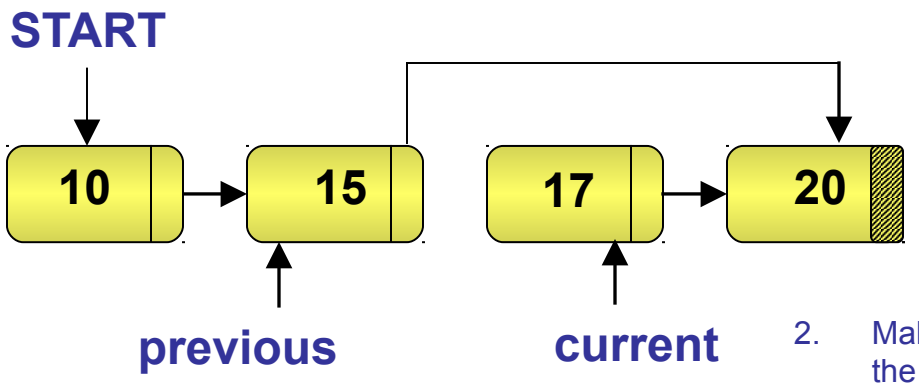
1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Previous -> next = current -> next

Deleting a Node Between two Nodes in the List (Contd.)

- Delete 17

Delete operation complete



1. Locate the node to be deleted. Mark the node to be deleted as current and its predecessor as previous. To locate current and previous, execute the following steps:
   a. Set previous = NULL
   b. Set current = START
   c. Repeat step d and e until either the node is found or current becomes NULL.
   d. Make previous point to current.
   e. Make current point to the next node in sequence.
2. Make the next field of previous point to the successor of current.
3. Release the memory for the node marked as current.

Previous -> next = current -> next

ALGORITHM TO DELETE A NODE FROM THE SPECIFIC POSITION

```
Algorithm DeleteAtSpec()
{
1. Enter the Location
2. Current = Start
3. Previous = NULL
4. If (Start == 0)
        4.1 Print "underflow“
   else If ( Location == 1)
        4.1 Start = Start -> next
        4.2 Current -> next = NULL
        4.3 Release the memory [ free (Current) ]
   else
        4.1 for (i=1 to Location-1)
            4.1.1 Previous = Current
            4.1.2 Current = Current -> next
        4.2 Previous -> next = Current -> next
        4.3 Current -> next = NULL
        4.4 Release the memory [ free (Current) ]
}
```

- Problem Statement
  - Discuss the advantages and disadvantages of linked lists.

- Problem Statement
  - Discuss the differences between arrays and linked lists.

- Linked lists allow _____ access to elements.

- Answer:
  - sequential