SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

Course Name: 23ITB201 Data structures and Algorithms

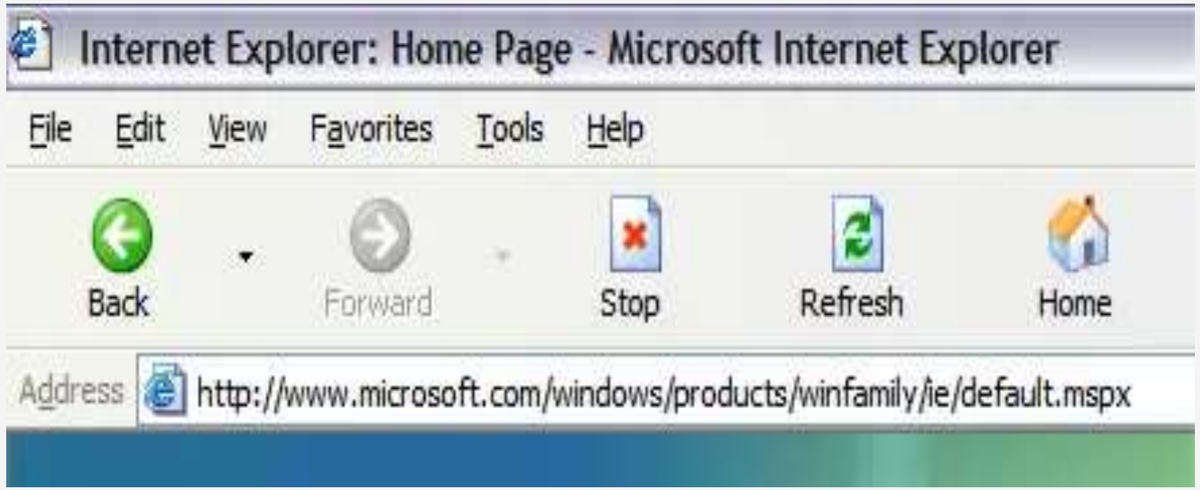
II Year / III semester

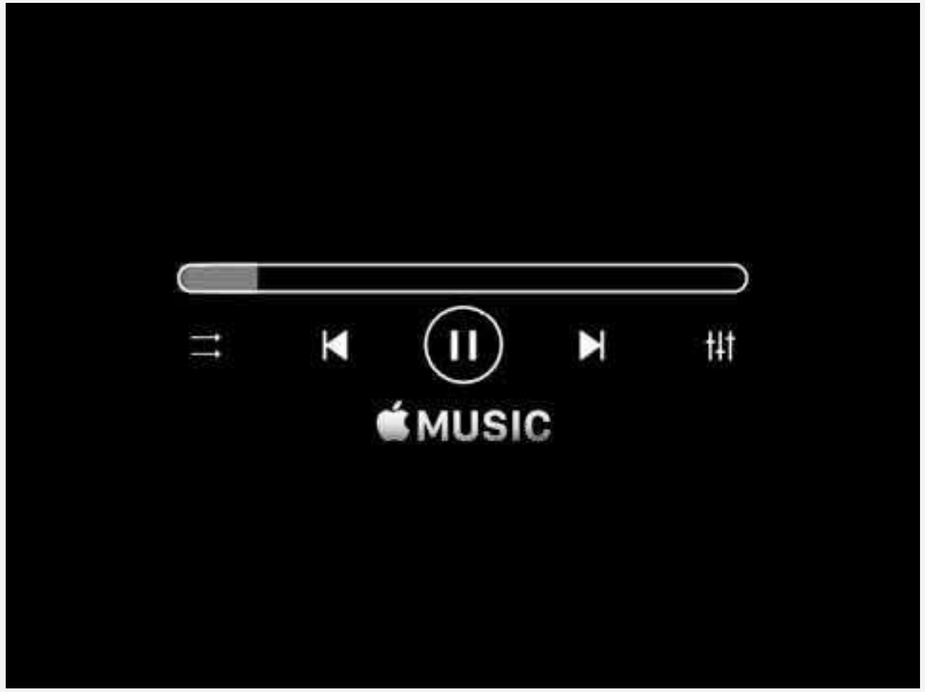
Unit I – List ADTs

Topic: Linked list

Linked list

Think !!!

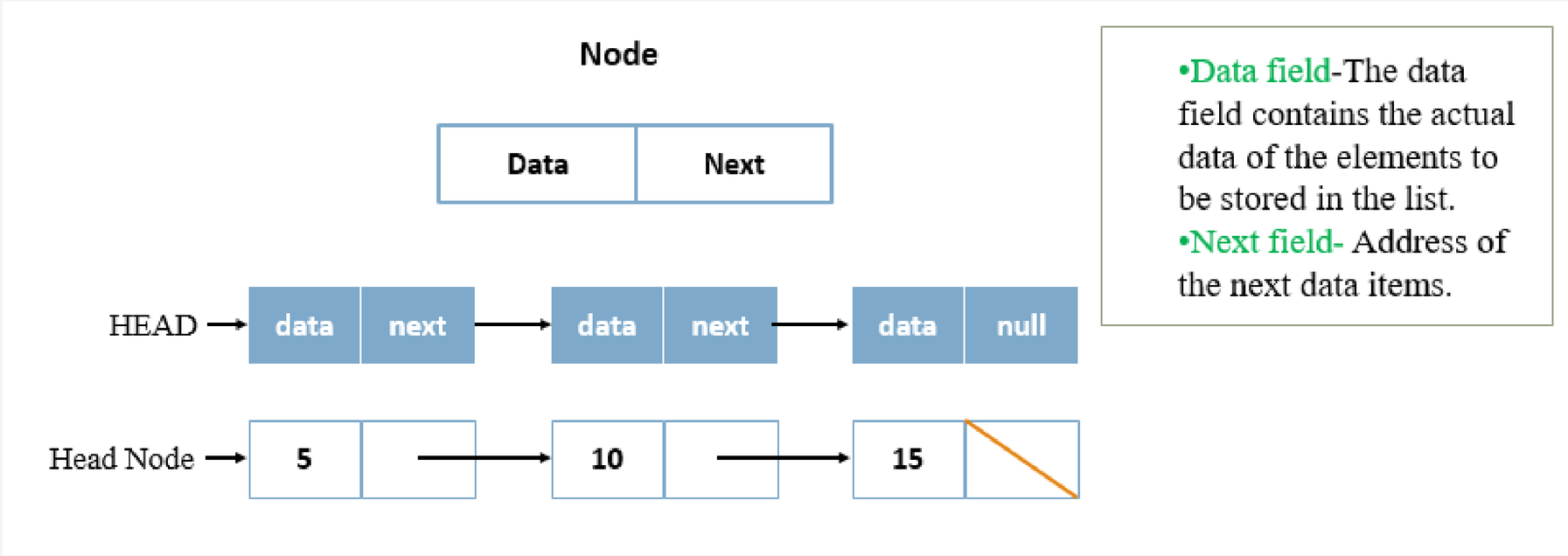

Internet Explorer: Home Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Home
Address
http://www.microsoft.com/windows/products/winfamily/ie/default.mspx


MUSIC

Linked List

- Linked list is an ordered collection of elements which are connected by **Links / pointers.**
- Memory is allocated at run time i.e. **dynamic memory allocation.**
- New elements can be stored anywhere in the memory (non contiguous memory location)
- Each node contains two fields namely,
- **Data field-**The data field contains the actual data of the elements to be stored in the list.
- **Next field-** Address of the next data items.

Linked List Representation



- Data field-The data field contains the actual data of the elements to be stored in the list.
- Next field- Address of the next data items.

Types of Linked List

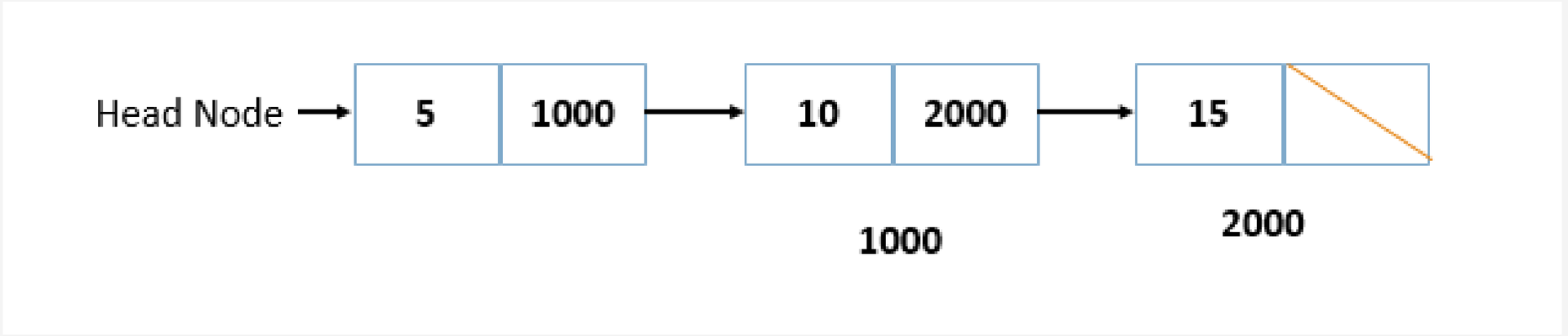
Types of Linked List

- Singly Linked List
- Doubly Linked List
- Circular Linked List

Singly Linked list

Singly Linked list

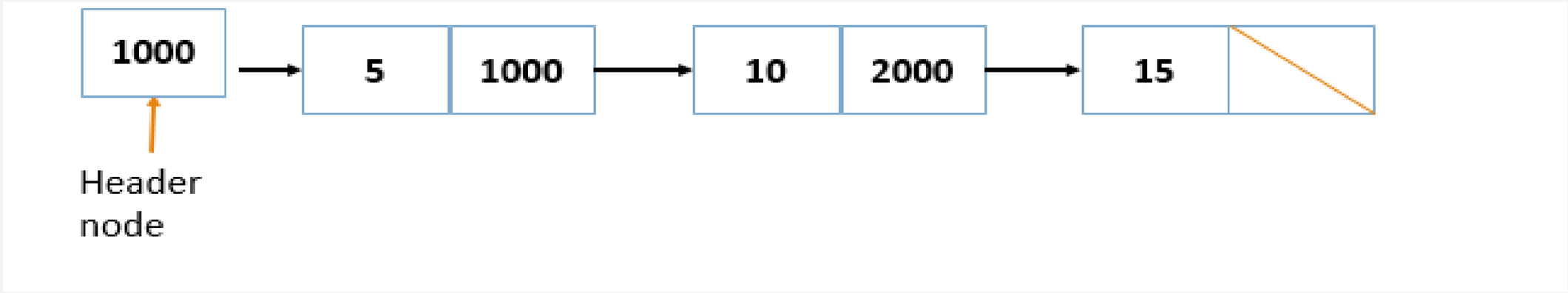
A singly linked list is a linked list in which each node contains only one link field pointing to the next node in the list.



Singly Linked list

Singly Linked list with Header

- The header node is at the very beginning of the linked list.
- It is an extra node kept at the front of a list.
- Does not represent an item in the linked list.
- It contains the address of the first node.



Operations on SLL

Basic operations on a singly-linked list are:

- Insert – Inserts a new node in the list.
- Delete – Deletes any node from the list.
- Display-display the date in the list
- Search-find whether a element is present in the list or not

Singly Linked List - Structure Definition

```
struct node
{
int data;
struct node * next;
};
```

Size of this Structure:
For Integer: 4 bytes
For Pointer : 4 Bytes
Totally : 8 bytes

malloc()

malloc():

malloc()

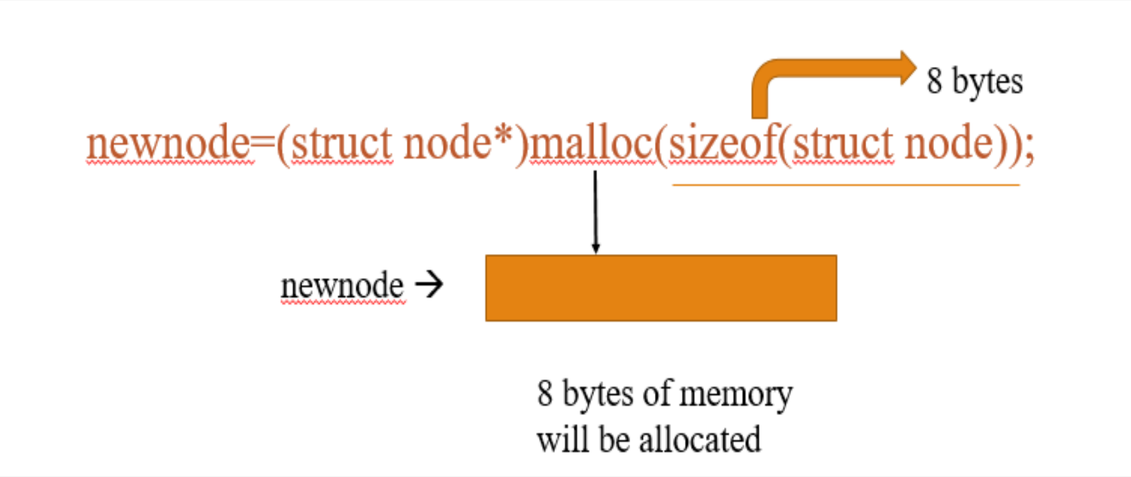
The malloc function reserves a block of memory of specified size and returns a pointer (starting address of the memory block).

The general syntax of malloc() is

ptr =(cast-type*)malloc(byte-size);

Example:

newnode=(struct node*)malloc(sizeof(struct node));



Storing data in the data field

To access members of a structure through a **pointer**, use the (->) arrow operator.

Example:

newnode -> data = 10;

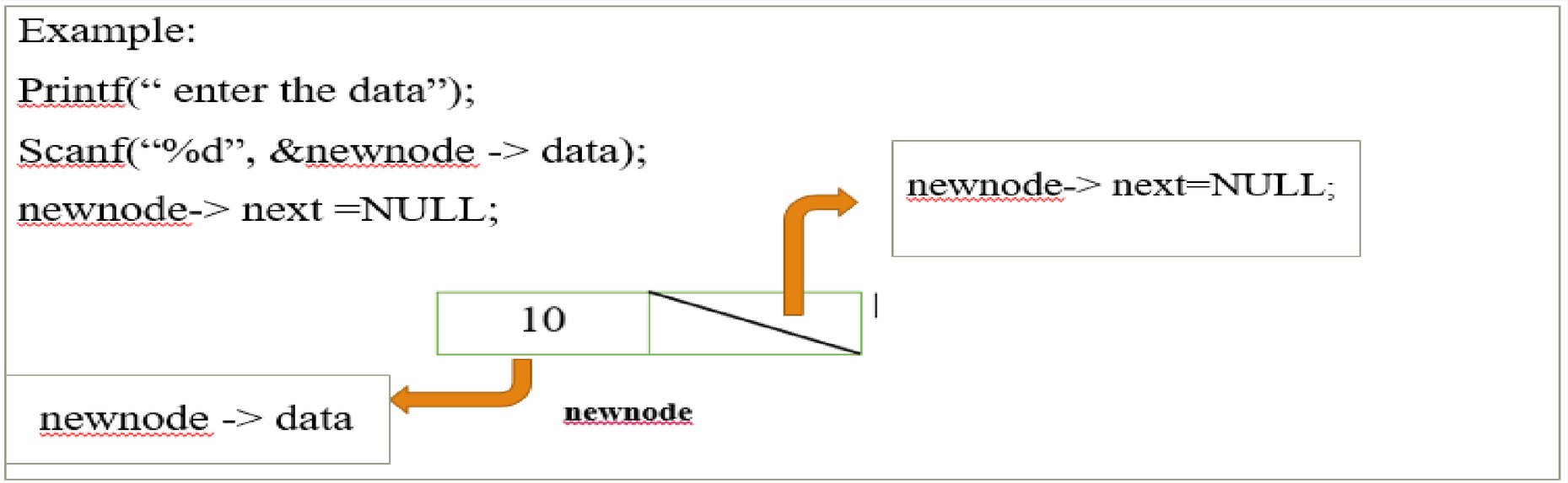
node data field of the node

Example:

Printf(" enter the data");

Scanf("%d", &newnode -> data);

newnode-> next =NULL;



Steps for writing code

Steps for writing code

1. Define Structure of the node
2. Allocate memory
3. Get or assign the values to the data field
4. Make Connection between the nodes.

Steps for writing code

Singly Linked List Insertion operation

There are three possible cases when we want to insert an element in the linked list-

- Insertion of a node as a head(first) node
- Insertion of a node as a last node
- Insertion of a node after some node

Insertion operation

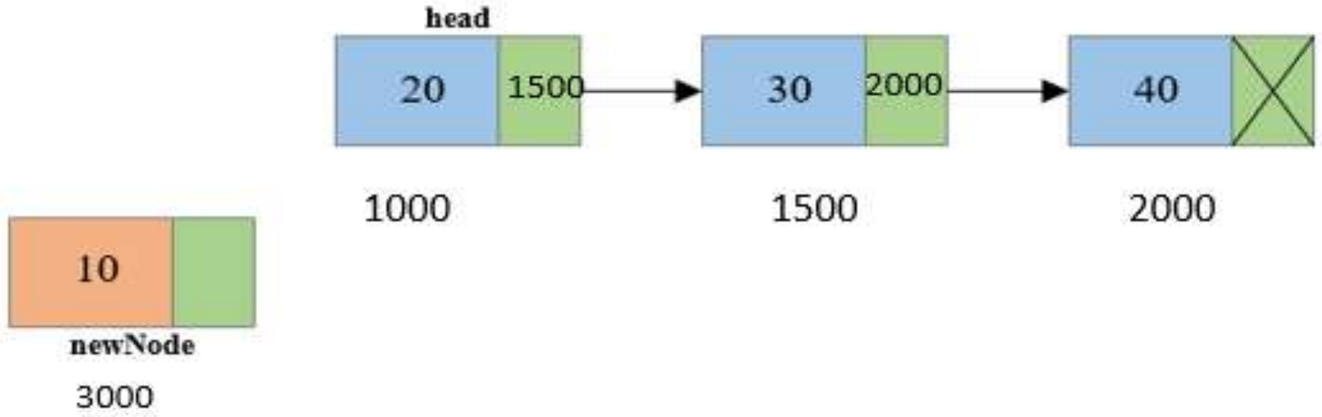
There are three possible cases when we want to insert an element in the linked list-

- Insertion of a node as a head(first) node
- Insertion of a node as a last node
- Insertion of a node after some node

Insert a node as first node

Steps to insert a node at the beginning of singly linked list

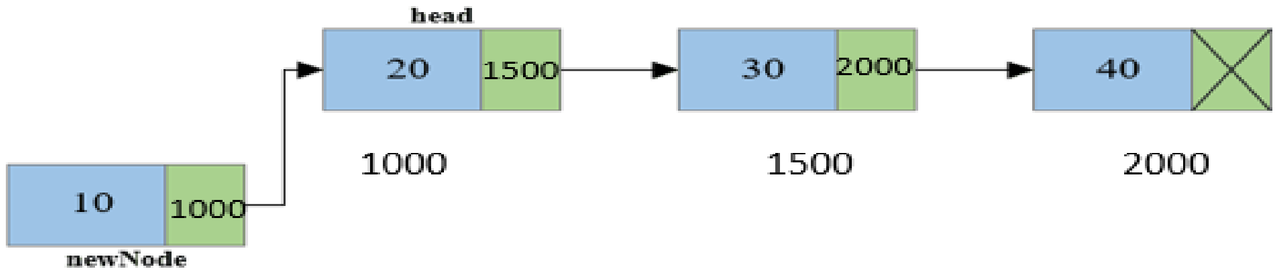
1. Create a new node, say newnode points to the newly created node.



Insert a node as first node

Steps to insert a node at the beginning of singly linked list

2. Link the newly created node with the head node, i.e. the newnode will now point to head node.

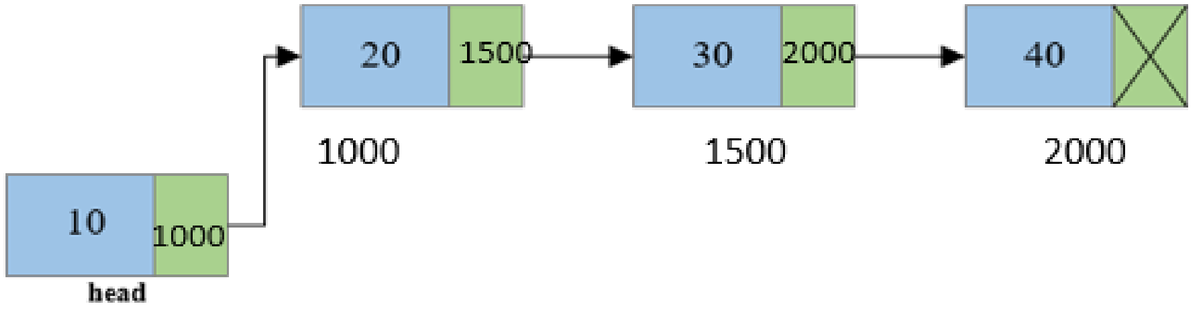



```
newnode -> next = head;
```

Insert a node as first node

Steps to insert a node at the beginning of singly linked list

3. Make the new node as the head node, i.e. now head node will point to newnode




```
head = newnode;
```

Insert a node as first node

Algorithm to insert a node as first node

Step 1. Create a new node and assign the address say newnode.

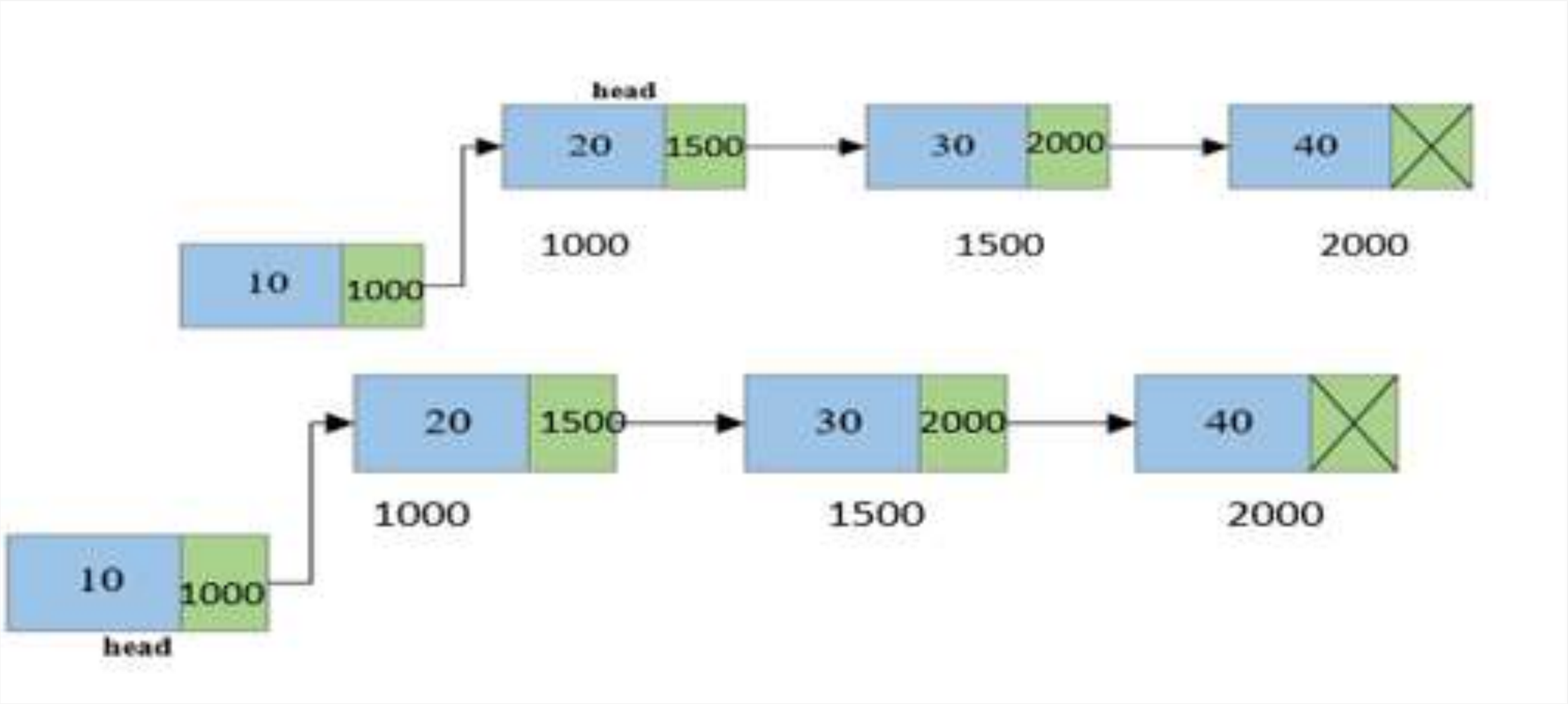
Step 2. Assign newnode -> data = value;

Step 3. Assign newnode -> next = head;

Step 4. Set head = newnode;

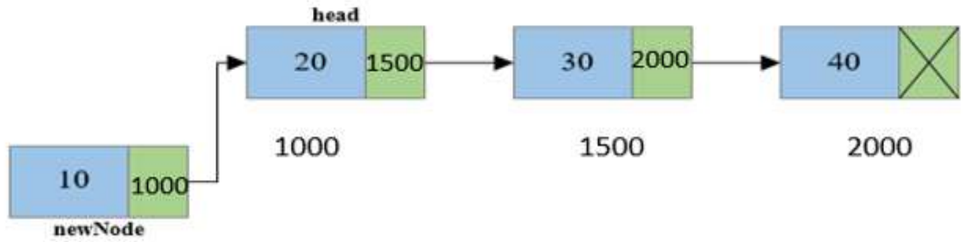
Step 5. EXIT

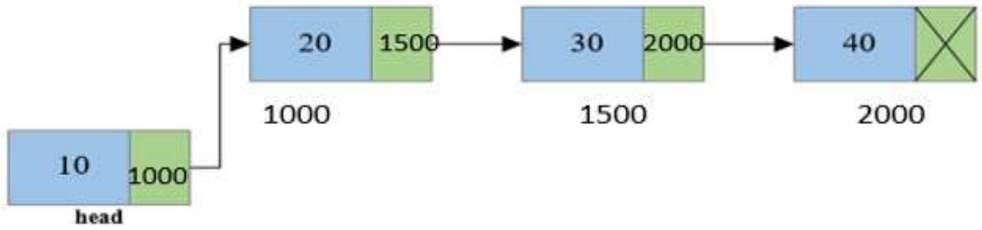
Insertion operation



C Routine to insert node at beginning of List

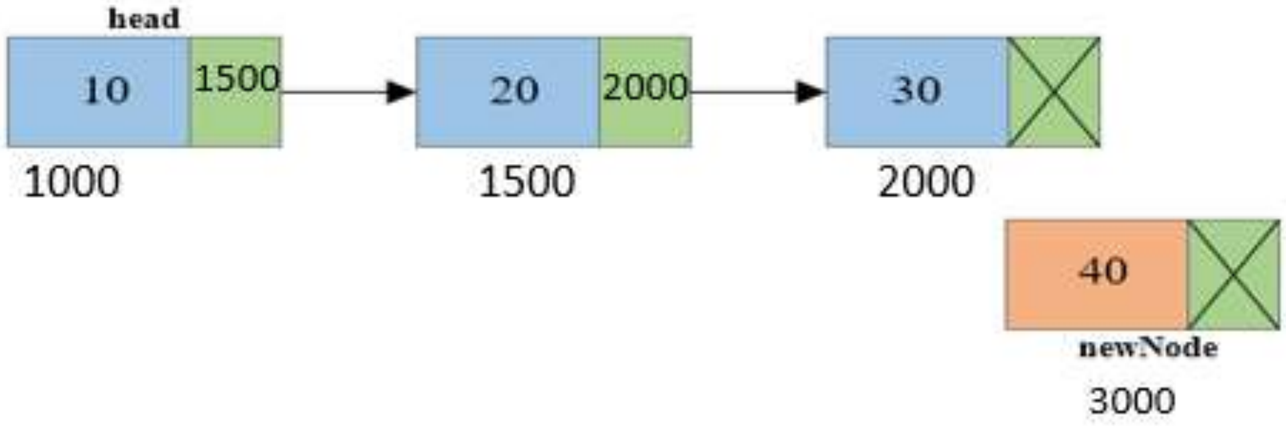
```
void insertNodeAtBeginning(int data
{
    struct node *newNode;

     newNode = (struct node*)malloc(sizeof(struct node));
     newNode->data = data; //  data part
      newNode->next = head; // Link address part
        head = newNode;  // Make newNode as first node

}
```





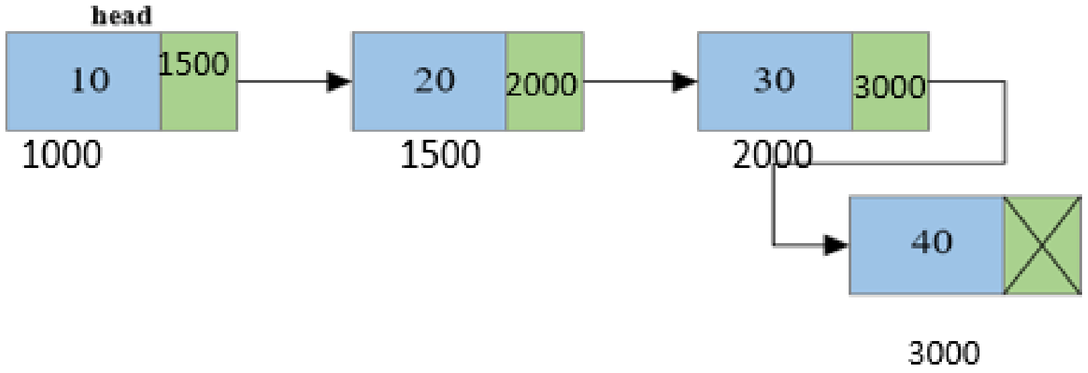
Inserting node at end

Create a new node and make sure that the address part of the new node points to NULL i.e. newNode->next=NULL



Inserting node at end

2. Traverse to the last node of the linked list and connect the last node of the list with the new node, i.e. last node will now point to new node. (temp->next = newNode).




```
temp -> next = newnode;
```

Algorithm to insert a node as last node

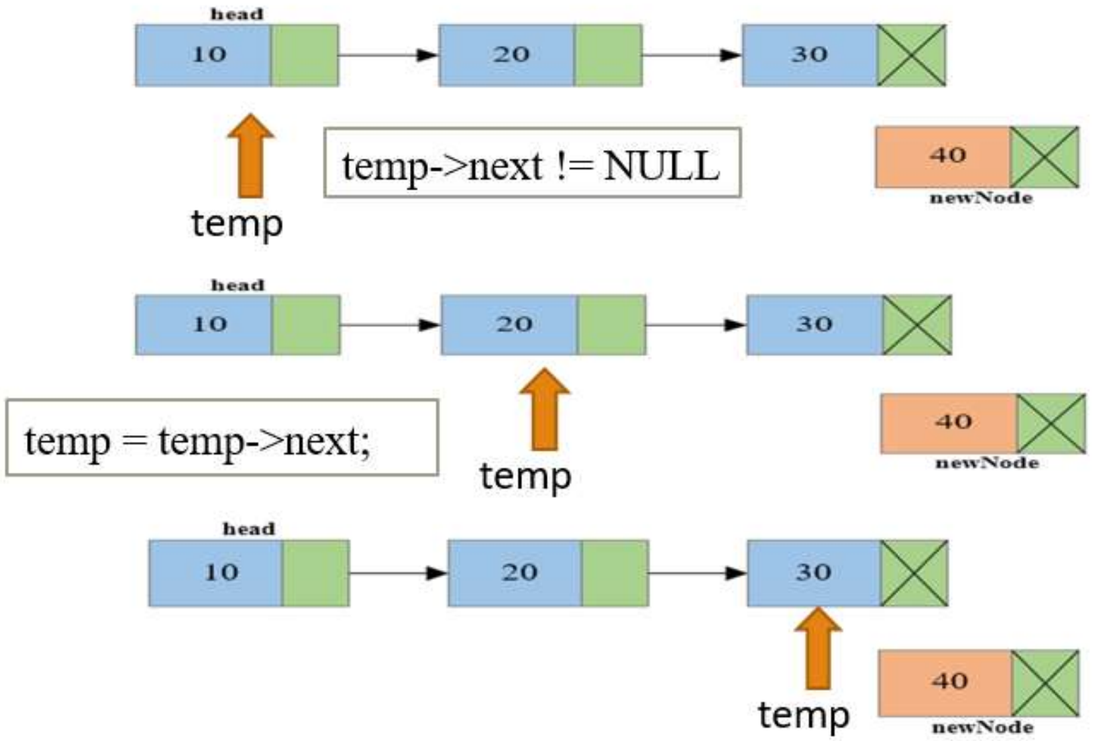
Algorithm to insert a node as last node

1. Create a node, say newnode.
2. Assign newnode -> data = value;
   Set newnode -> next = NULL;
3. Set temp = head;

   WHILE temp->next != NULL

   temp = temp->next;

   temp->next = newnode;



Algorithm to insert a node as last node

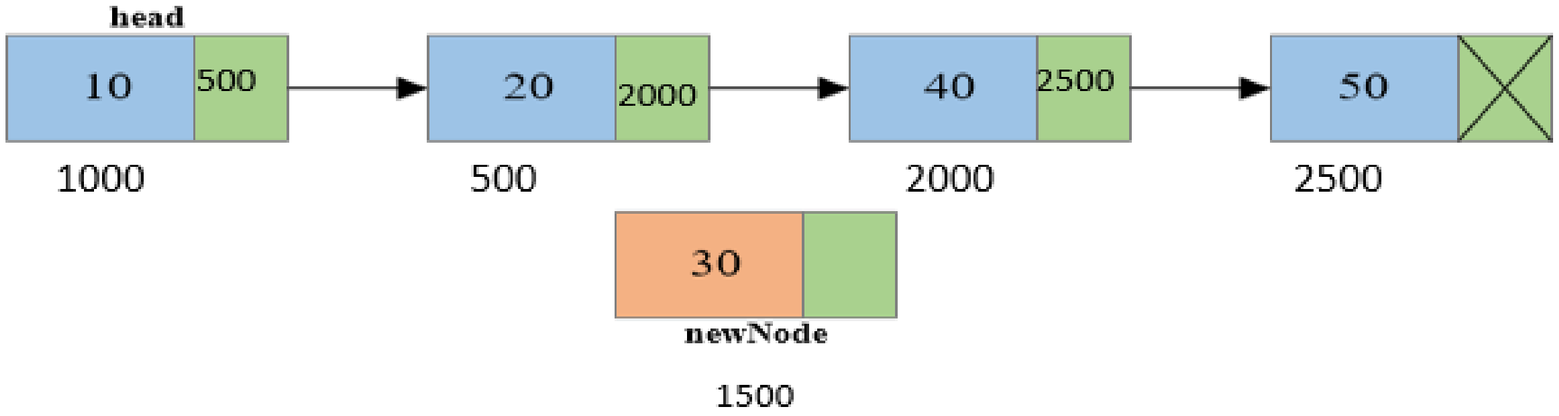
C Routine to insert a node as last node

```
void insertNodeAtEnd(int data)
{
    struct node *newNode, *temp;
    newNode = (struct node*)malloc(sizeof(struct node));
        newNode->data = data;  //data part
        newNode->next = NULL;
        temp = head;
   // Traverse to the last node
        while(temp->next != NULL)
            temp = temp->next;
        temp->next = newNode; // Link address part
}
```

Insert a node at any intermediate position based on key value of singly linked list

Steps to insert a node at any intermediate position based on key value of singly linked list

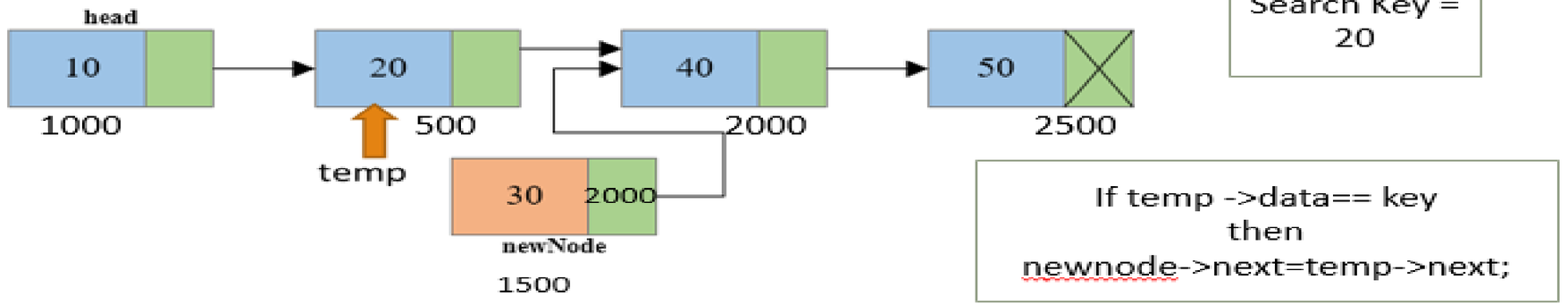
1. Create a new node say newnode



Insert a node at any intermediate position based on key value of singly linked list

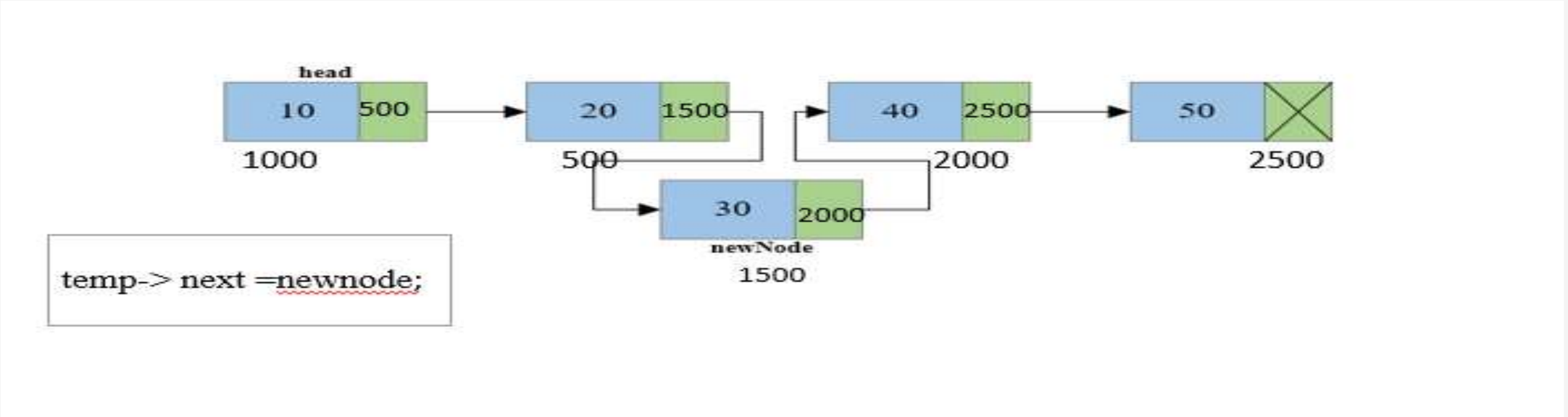
Steps to insert a node at any intermediate position based on key value of singly linked list

2. Get the search key value from user and compare each node's data.



Insert a node at any intermediate position based on key value of singly linked list

Steps to insert a node at any intermediate position based on key value of singly linked list



Insert a node at any intermediate position based on key value of singly linked list

Algorithm to insert a node at any intermediate position based on key value of singly linked list

```
Create a node, say newnode.
Assign newnode -> data = value;
Get the search key value as key
        set temp = head;
    Do
    If temp->data == key;
       newnode-> next=temp->        next;
       temp->next= newnode;
     Else
        temp = temp->next;
 WHILE temp->next != NULL
```

Deletion operation in SLL

Deletion operation in SLL

Possible cases when we want to delete an element in the linked list-

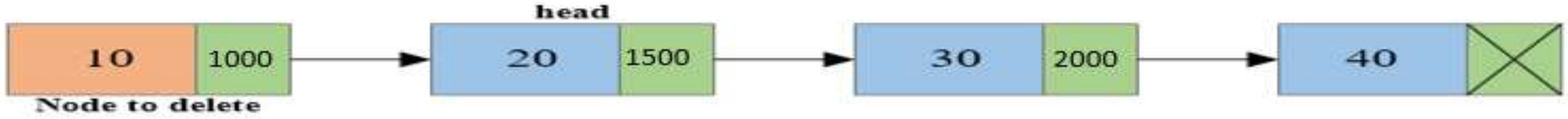
- Deleting the first node in the list.
- Deleting the last node in the list.
- Delete the list.
- Delete the node using key value

Deletion operation in SLL

Steps to delete first node from Singly Linked List

1. Copy the address of first node i.e. `head` node to some temp variable say `toDelete` .

2. Move the `head` to the second node of the linked list i.e. `head = head->next` .

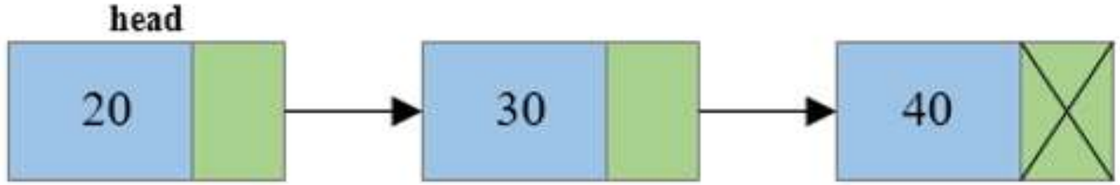


3. Disconnect the connection of first node to second node.

Deletion operation in SLL

Steps to delete first node from Singly Linked List

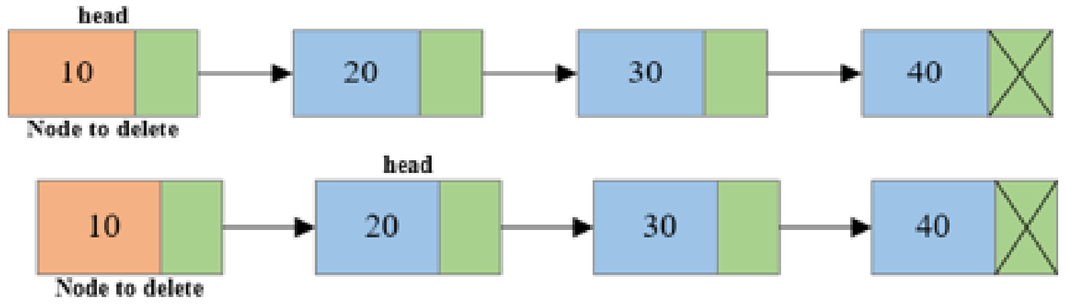
4. Free the memory occupied by the first node.



Deletion operation in SLL

C Routine to delete the first node in the list.

```
void deleteFirstNode()
{
   struct node *toDelete;
   toDelete = head;
   head = head->next;
   printf("\nData deleted = %d\n", toDelete->data);
      /* Clears the memory occupied by first node*/
   free(toDelete);

}
```

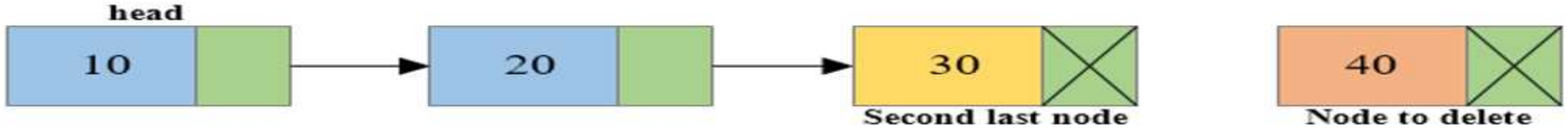


Deletion operation in SLL

Steps to delete last node of a Singly Linked List

1. Traverse to the last node of the linked list keeping track of the second last node in some temp variable say `secondLastNode` .

2. If the last node is the `head` node then make the head node as `NULL` else disconnect the second last node with the last node i.e. `secondLastNode->next = NULL` .



3. Free the memory occupied by the last node.

Deletion operation in SLL

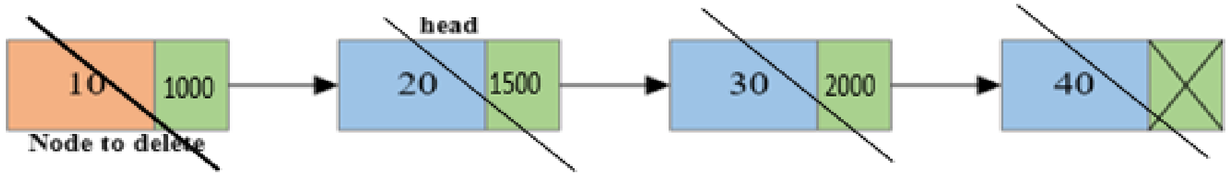
C Routine to delete the last node

```
void deleteLastNode(){
struct node *toDelete,
*secondLastNode;
toDelete = head;
secondLastNode = head;
 /* Traverse to the last node of the
list */
        while(toDelete->next != NULL)
         {
 secondLastNode = toDelete;
          toDelete = toDelete->next;}
   if(toDelete == head)
   { head = NULL;
   }
   else
   {
        /* Disconnect */
secondLastNode->next = NULL;
   }
     free(toDelete);      }
```

Deletion operation in SLL

C Routine to delete all nodes of Singly Linked List

```
void deleteList()
{
     struct node *temp;

     while(head != NULL)
     {
          temp = head;
          head = head->next;

          free(temp);
     }

}
```



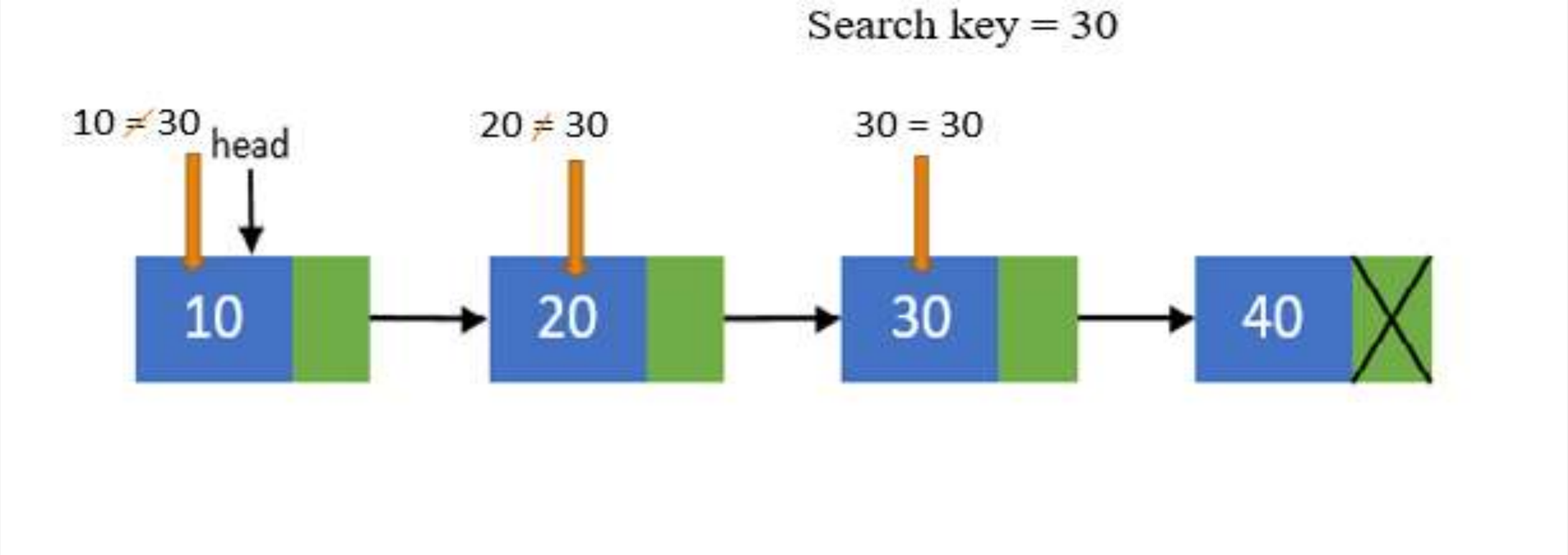
Deletion operation in SLL

Delete by key value

```
void deleteFirstByKey(int key)
{
    struct node *prev, *cur;

    /* Check if head node contains key */
    while (head != NULL && head->data ==key)
    {
        // Get reference of head node
        prev = head;
         head = head->next;

       free(prev);

        // No need to delete further
        return;
    }
prev = NULL;
cur  = head;

  while (cur != NULL)
{
    // Current node contains key
    if (cur->data == key)
    {
          if (prev != NULL)
            prev->next = cur->next;
              free(cur);
        // No need to delete further
        return;
    }
```

Searching operation in Singly Linked List

Searching for a particular value in the Linked list from the Beginning.



Searching operation in Singly Linked List

C routine for searching

```
int search(struct node *head, int key)
{
    struct node * temp;
    temp = head;
    while (temp != NULL)
    {
        if (temp->data == key)
        {
            return 1;
        }
        temp = temp->next;
    }
    return 0 ;
}
```

Disadvantages of Singly Linked List

Disadvantages of Singly Linked List

- Traversing in reverse is not possible in case of Singly linked list.
- Singly linked list deletion requires a pointer to the node and previous node to be deleted.

Assessment

- How do you allocate memory for the node in C ?
- Create a structure for SLL node.


thank
you