Reverse linked List

Reverse linked list is a linked list created to form a linked list by reversing the links of the list. The head node of the linked list will be the last node of the linked list and the last one will be the head node.

Example

9 -> 32 -> 65 -> 10 -> 85 -> NULL

Example

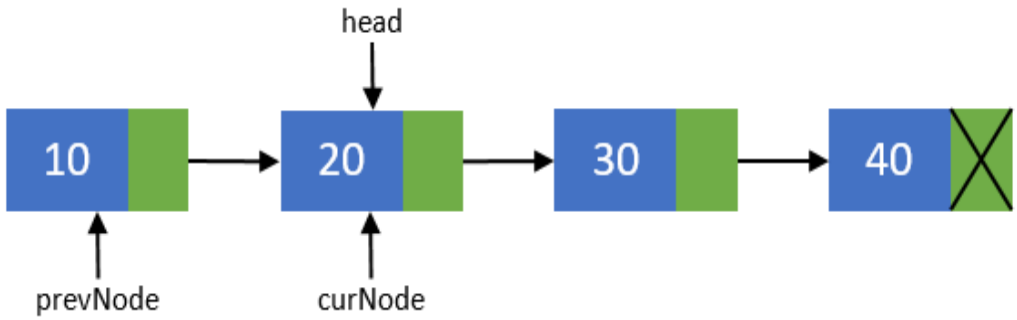
Reverse linked list formed from the above linked list −
85 -> 10 -> 65 -> 32 -> 9 -> NULL

Algorithm to reverse a Singly Linked List

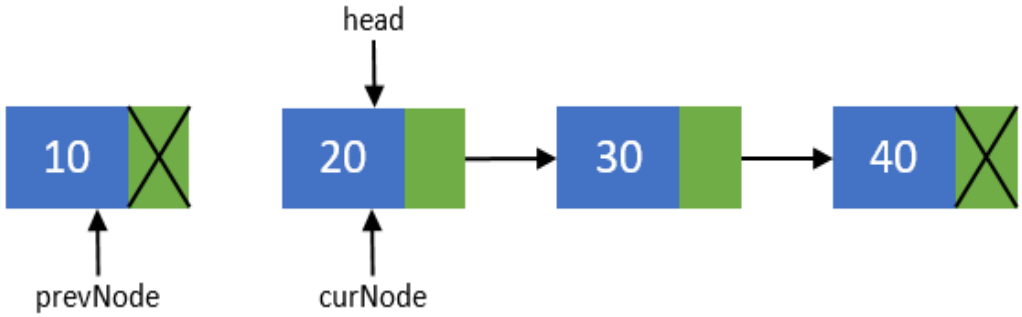
Algorithm to reverse a Singly Linked List
Input : ***head*** node of the linked list
Begin:
  If (*head* != **NULL**) then
    prevNode ← *head*
    head ← *head.next*
    curNode ← *head*
    prevNode.next ← **NULL**
    While (*head* != **NULL**) do
      head ← *head.next*
      curNode.next ← *prevNode*
      prevNode ← *curNode*
      curNode ← *head*
    End while
    head ← *prevNode*
  End if
End

Steps to reverse a Singly Linked List

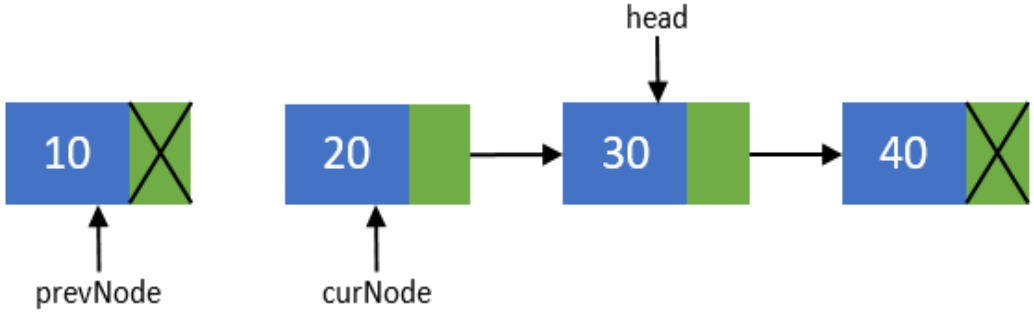
1. Create two more pointers other than head namely prevNode and curNode that will hold the reference of previous node and current node respectively.
Make sure that prevNode points to first node i.e. prevNode = head.
head should now point to its next node i.e. the second node head = head->next.
curNode should also points to the second node i.e. curNode = head.



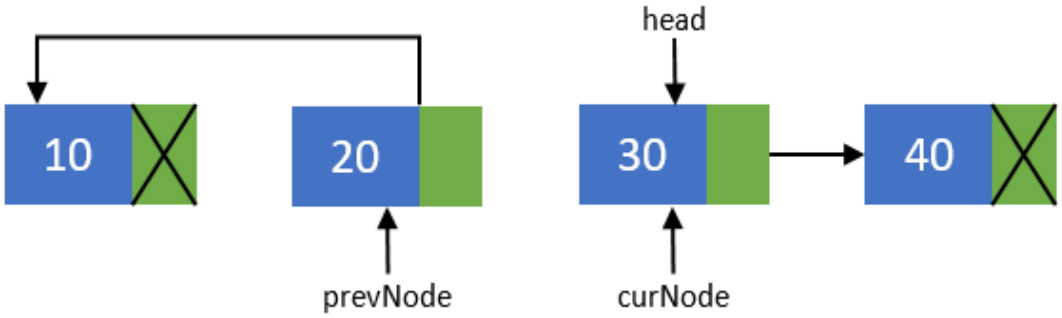
2. Now, disconnect the previous node i.e. the first node from others. We will make sure that it points to none. As this node is going to be our last node. Perform operation prevNode->next = NULL.



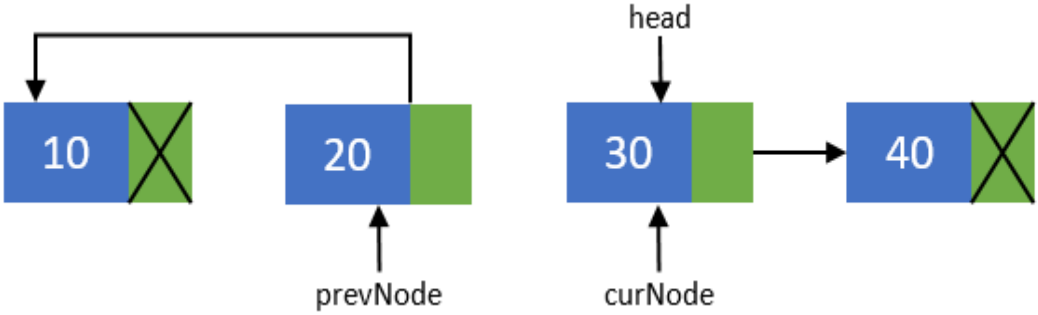
3. Move head node to its next node i.e. head = head->next.



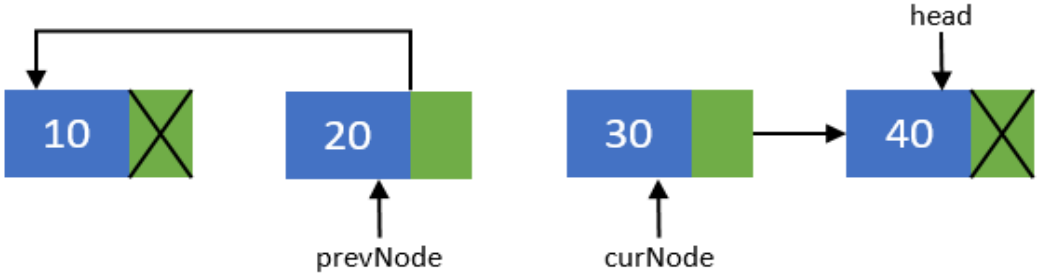
4. Now, re-connect the current node to its previous node i.e. curNode->next = prevNode;.

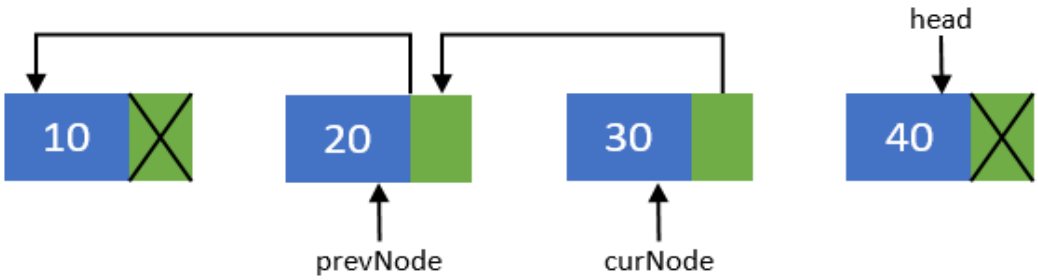


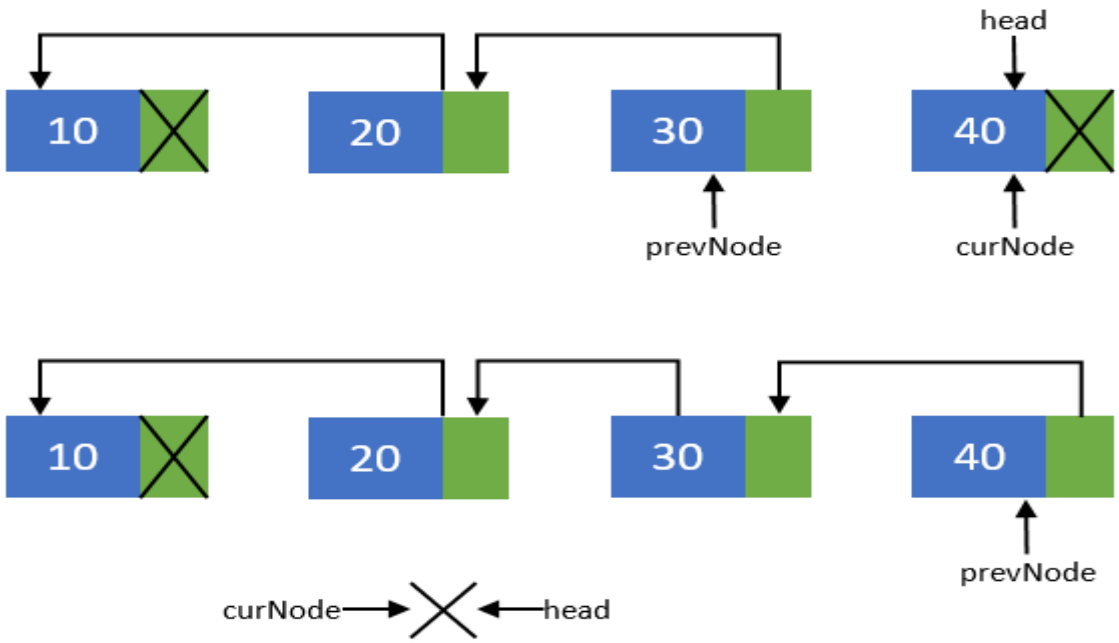
5. Point the previous node to current node and current node to head node. Means they should now point to prevNode = curNode; and curNode = head.



6. Repeat steps 3-5 till head pointer becomes NULL.







7. Now, after all nodes has been re-connected in the reverse order. Make the last node as the first node. Means the **head** pointer should point to prevNode pointer. Perform head = prevNode;. Finally you end up with a reversed linked list of its original.

Example :

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
   int data;
   struct node *next;
}*head;
void createList(int n);
void reverseList();
void displayList();
```

```
int main()
{
   int n, choice;
   printf("Enter the total number of nodes: ");
   scanf("%d", &n);
   createList(n);

   printf("\nData in the list \n");
   displayList();

   printf("\nPress 1 to reverse the order of singly linked list\n");
   scanf("%d", &choice);
   if(choice == 1)
   {
      reverseList();
   }

   printf("\nData in the list\n");
   displayList();

   return 0;
}
void createList(int n)
{
   struct node *newNode, *temp;
   int data, i;

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

   if(head == NULL)
   {
      printf("Unable to allocate memory.");
   }
```

```
    else
    {
        printf("Enter the data of node 1: ");
        scanf("%d", &data);

        head->data = data;
        head->next = NULL;

        temp = head;

        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));
            if(newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data;
                newNode->next = NULL;

                temp->next = newNode;
                temp = temp->next;
            }
        }
        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}
```

```
void reverseList()
{
   struct node *prevNode, *curNode;

   if(head != NULL)
   {
      prevNode = head;
      curNode = head->next;
      head = head->next;
      prevNode->next = NULL;
      while(head != NULL)
      {
         head = head->next;
         curNode->next = prevNode;

         prevNode = curNode;
         curNode = head;
      }
      head = prevNode;
      printf("SUCCESSFULLY REVERSED LIST\n");
   }
}
void displayList()
{
   struct node *temp;
   if(head == NULL)
   {
      printf("List is empty.");
   }
   else
   {
      temp = head;
      while(temp != NULL)
```

```
    {
        printf("Data = %d\n", temp->data);
        temp = temp->next;
    }
  }
}
```

Output

```
Enter the total number of nodes: 5
Enter the data of node 1: 10
Enter the data of node 2: 20
Enter the data of node 3: 30
Enter the data of node 4: 40
Enter the data of node 5: 50
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 10
Data = 20
Data = 30
Data = 40
Data = 50
```

Output

```
Data = 20
Data = 30
Data = 40
Data = 50

Press 1 to reverse the order of singly linked list
1
SUCCESSFULLY REVERSED LIST

Data in the list
Data = 50
Data = 40
Data = 30
Data = 20
Data = 10
```