**[finding merge point in linked list using traversal !!](http://chanduthedev.blogspot.in/2012/07/finding-merge-point-in-linked-list.html)**

Below is the C code for finding the merge point in two linked lists. [Click here](http://chanduthedev.blogspot.in/2012/07/linked-list-intersection.html) for explanation.

[?](http://chanduthedev.blogspot.in/2012/07/finding-merge-point-in-linked-list.html)

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  struct node  {   int info;   int visited;   struct node \*next;  };    typedef struct node Node;    Node\* append(Node \*h, int info)  {   Node \*tempHead=h;   Node \*newNode = (Node\*)malloc(sizeof(Node));   newNode->info = info;   newNode->visited = 0;   newNode->next = NULL;   if(tempHead == NULL) // if the list has zero elements. make new node as a head   {    h=newNode;   }   else if(tempHead->next==NULL)  // if the list is having only one node   {    tempHead->next = newNode;   }   else   {    Node \*tempNode = h;    while(tempNode->next != NULL)  // if the list has more than one node, so moving to the last node    {     tempNode = tempNode->next;    }    tempNode->next = newNode; // appending the new node at the end   }   return h;  }      /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  for displaying the nodes in the list  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  void display(Node \*h)  {   Node \*temp = h;   while (temp->next!=NULL)   {    printf("%d->",temp->info);    temp = temp->next;   }   printf("%d\n",temp->info);  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  this method visits/traverse each node from head and makes the visited flag true.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    void traverse(Node \*head)  {   Node \*tmp = head;   while(tmp)   {    //making the visited flag true    tmp->visited = 1;    tmp=tmp->next;   }  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  first traverse List one and make each node visited flag as true. Then start  traversing the second list, and check visited flag in each node, if flage is true  that node is the merge node.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  Node\* get\_merge\_node(Node \*h1, Node\* h2)  {   traverse(h1); // traversing first list and making the visit flag true.   while(h2!=NULL)   {    if(h2->visited)     return h2;    h2=h2->next;   }   return NULL;  }      main()  {   Node \*head1 = NULL;   Node \*head2 = NULL;   Node \*merge\_node = NULL;   int i;   for (i=1;i<=10;i++)   {    head1 = append(head1,i\*10);   }   for (i=1;i<=5;i++)   {    head2 = append(head2,i\*2);   }   Node \*temp1 = head1;   for (i=1;i<=8;i++)   {    temp1 = temp1->next;   }   printf("List one elements ....\n");   display(head1);   printf("List two elements ....\n");   display(head2);     Node \*temp2 = head2;   while(temp2->next!=NULL)    temp2=temp2->next;   //making merge point for two linked lists, commet below line for remvoing the merge point   temp2->next = temp1;   printf("List two elements after making merge point ....\n");   display(head2);   merge\_node = get\_merge\_node(head1,head2);   if(merge\_node)    printf("merge point data is %d\n",merge\_node->info);   else    printf("two lists are different\n");  } |