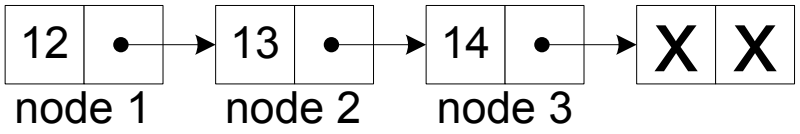
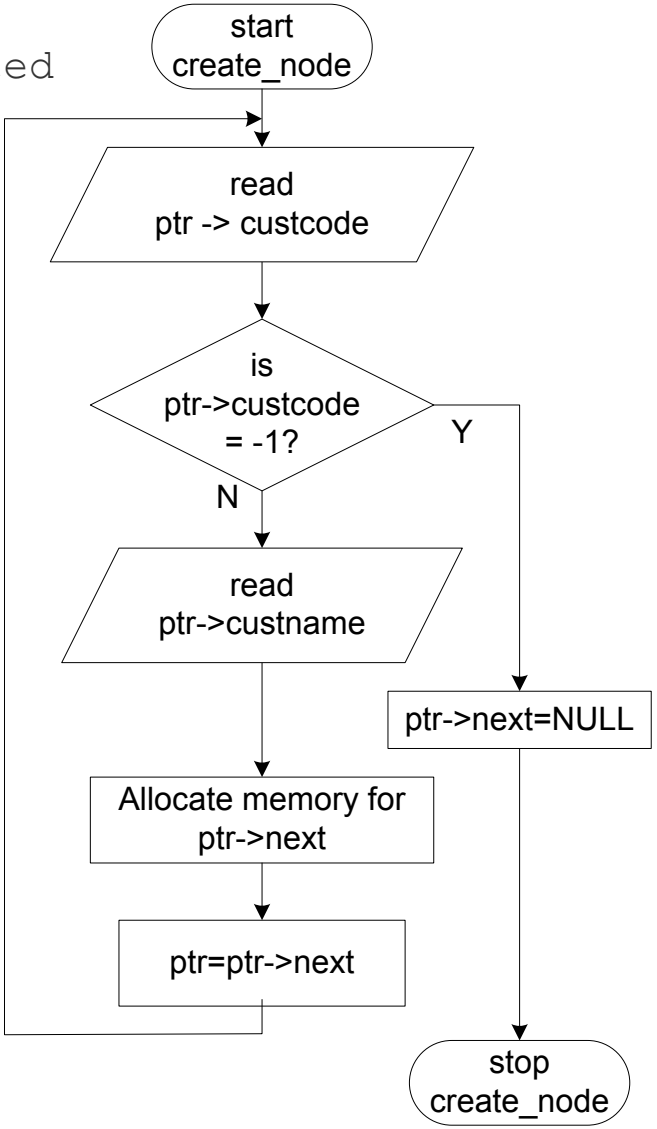
An algorithm is a step by step procedure describing the solution of a given problem. The following algorithm describes creatindisplaying and deleting an algorithm.

http://hubpages.com/hub/C-program-code-for-linked-list-manipulations

Algorithm To Create a Node In A Linked List

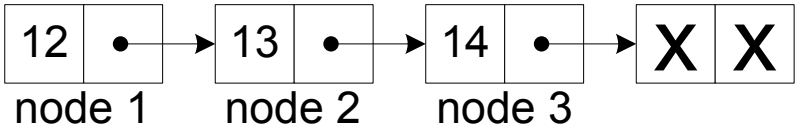
```
Step 1: Pass the pointer ptr where a node will be created
        void nodecreate(node * ptr)

Step 2: Print a message to enter code and press return

Step 3: Read ptr->custcode from the keyboard

Step 4: if((ptr->custcode)!=-1)
      print a message to entewr customer name
      read customer name (ptr->custname)from keyboard
      allocate memory size of node at ptr->next
      call nodecreate(ptr->next)again

Step 5: othervise
       assign NULL to ptr->next and finish
```

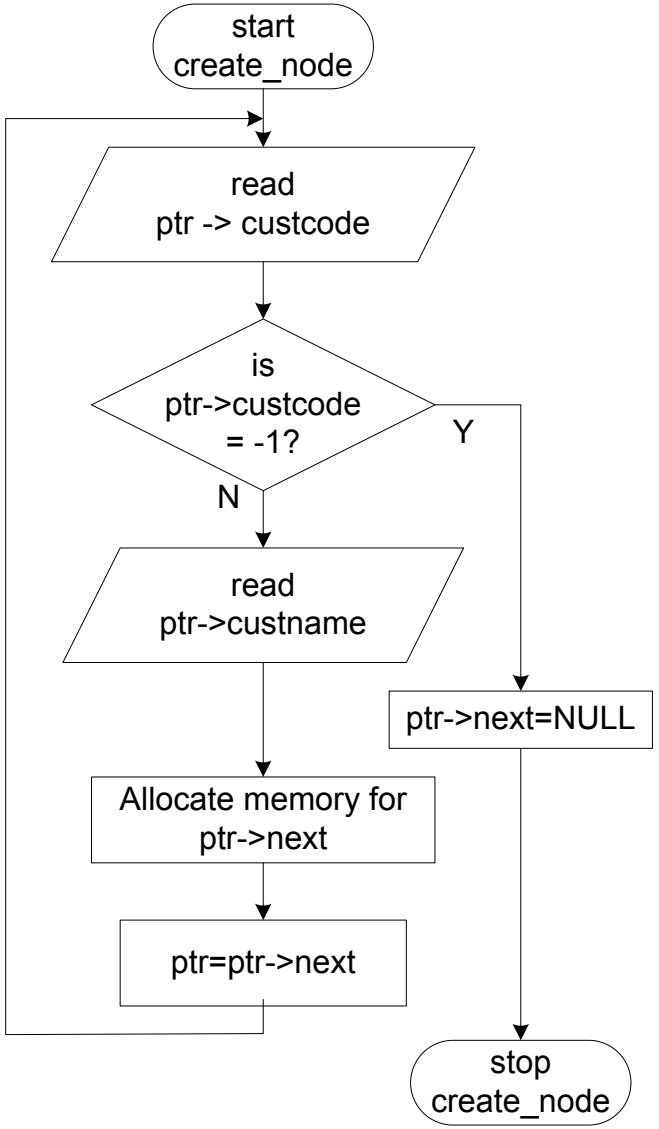




C Program to Cretae a Node

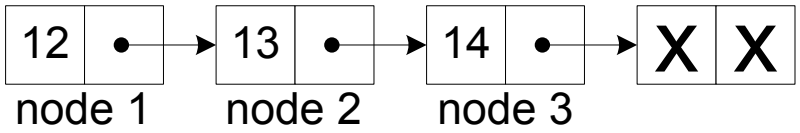
```
void nodecreate(node * ptr)
{
  printf("\n\t\tEnter code and press enter:");
  scanf("%d", &ptr->custcode);
  if((ptr->custcode)!=-1)
  {
    printf("\n\t\tEnter customer name:");
    scanf("%s", ptr->custname);
    ptr->next=(node *)malloc(sizeof(node));
    nodecreate(ptr->next);
    }
    Else
    ptr->next=NULL;
    }
```

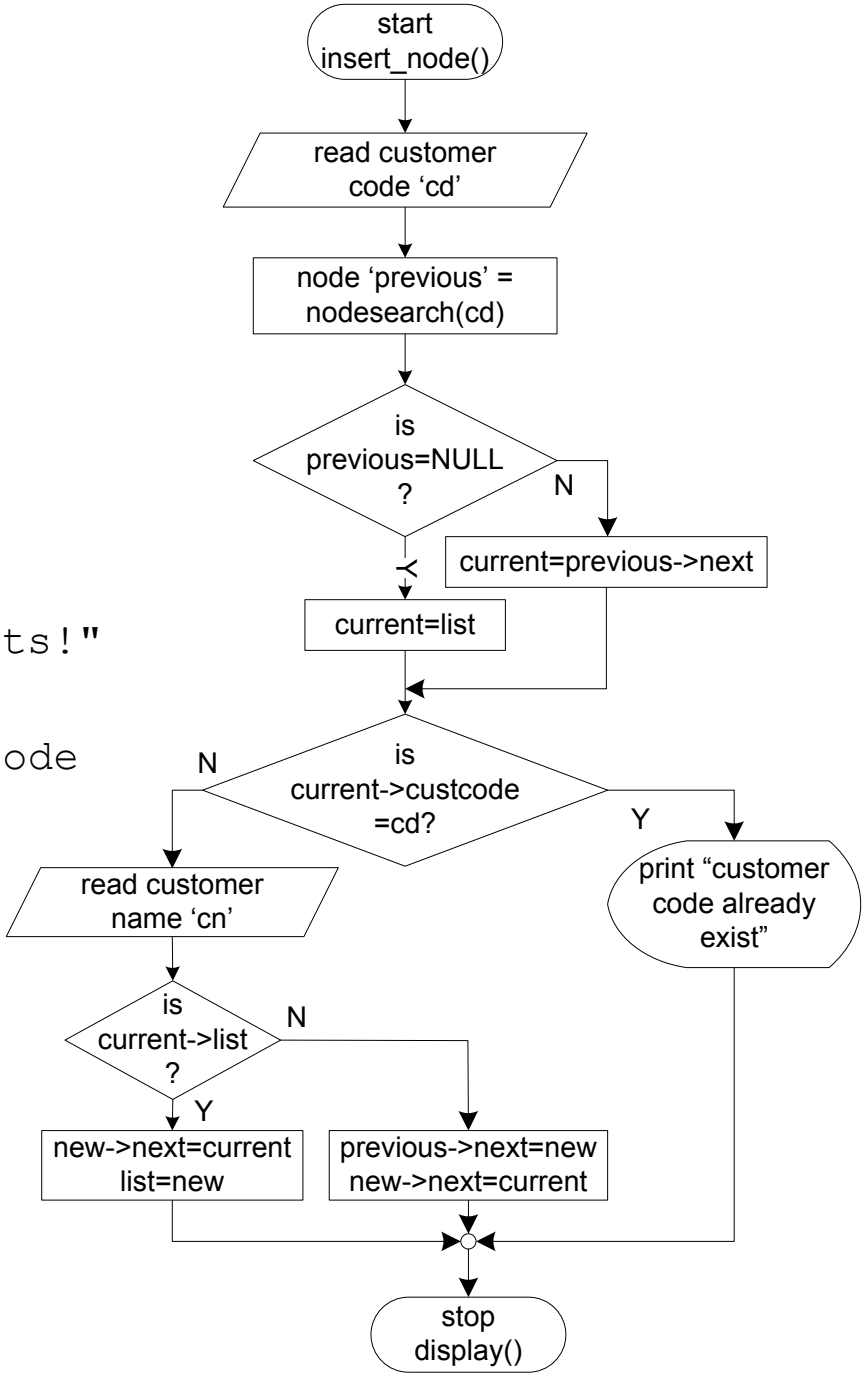




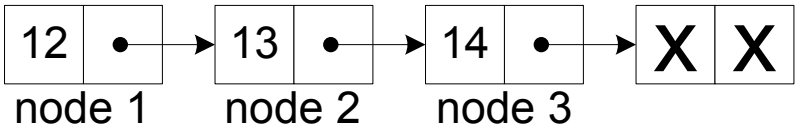
Algorithm insert_node to a linked list

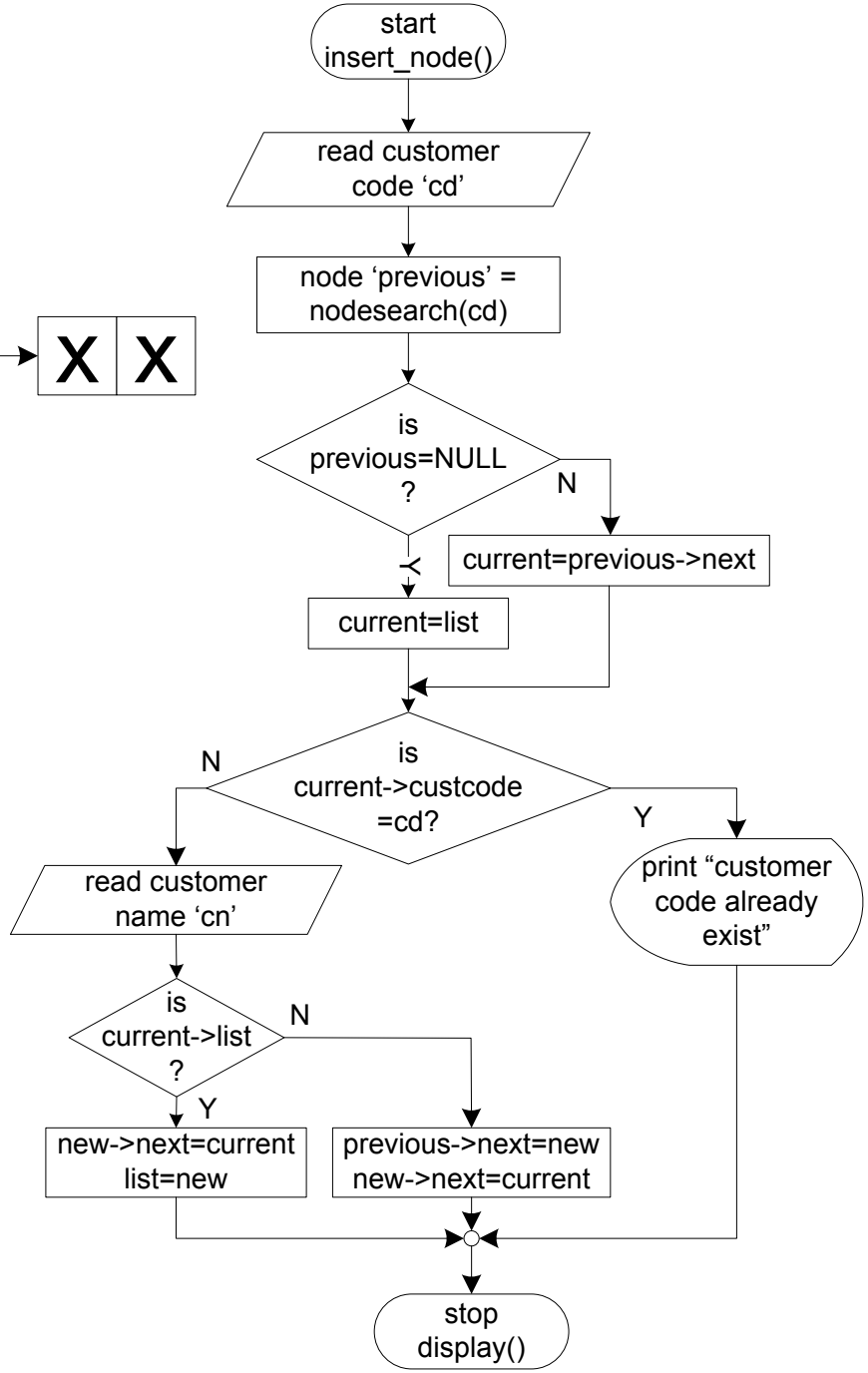
```
Step 1: read the customer code

Step 2: search if the node already exists

Step 3: check if previous is NULL
        or assign next to current
           if(previous==NULL)
              current=list;
           else
              current=previous->next;

Step 4: print error mesage if the node
        already exists
           if((current->custcode)==cd)
              print "Customer code already exists!"
           else
              print "Enter new customername:"
           assign new code cd to new costumer code

Step 5: if current node is listed, assign
        it to new->next and new to list,
        othervide make following assignment
           if(current==list)
              new->next=current;
              list=new;
           else
              previous->next=new;
              new->next=current;
```





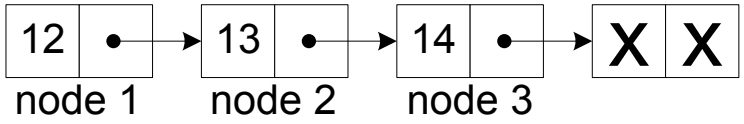
```
void insertnode(node * new)
{
  int cd;
  char *cn;
  node *previous, *current;

  printf("\nEnter the customercode to insert:");
  scanf("%d", &cd);

  prev=nodesearch(cd);

  if(previous==NULL)
    current=list;
  else
    current=previous->next;

  if((current->custcode)==cd)
  {
    printf("\nCustomer code already exists!!");
    getch();
  }
  else
  {
    printf("\n\t\tEnter customername:");
    scanf("%s", cn);
    new->custcode=cd;
    strcpy(new->custname,cn);

    if(current==list){
      new->next=current;
      list=new;
    }
    else{
      prev->next=new;
      new->next=cur;
    }
  }
}
```



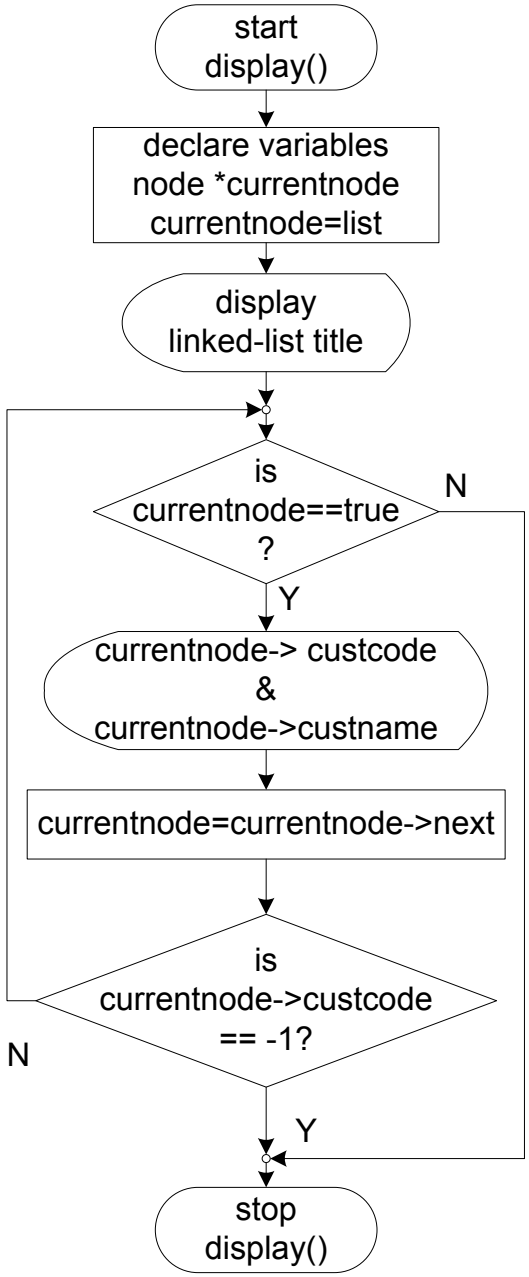


Algorithm To Display The Nodes In A Linked List

```
Step 1: Declare *currentnode to be of type node
Step 2: Assign the contents of list to currentnode
Step 3: Print an informing message "The Linked List is:"
Step 4: Print "Customer Code" and "Costomer Name"
Step 5: While currentnode is true, do untill step 7
          Print currentnode -> custcode and
          Print currentnode -> custname
Step 6: Assign currentnode->next to currentnode
Step 7: if currentnode -> custcode equals -1
        break the while loop…
```

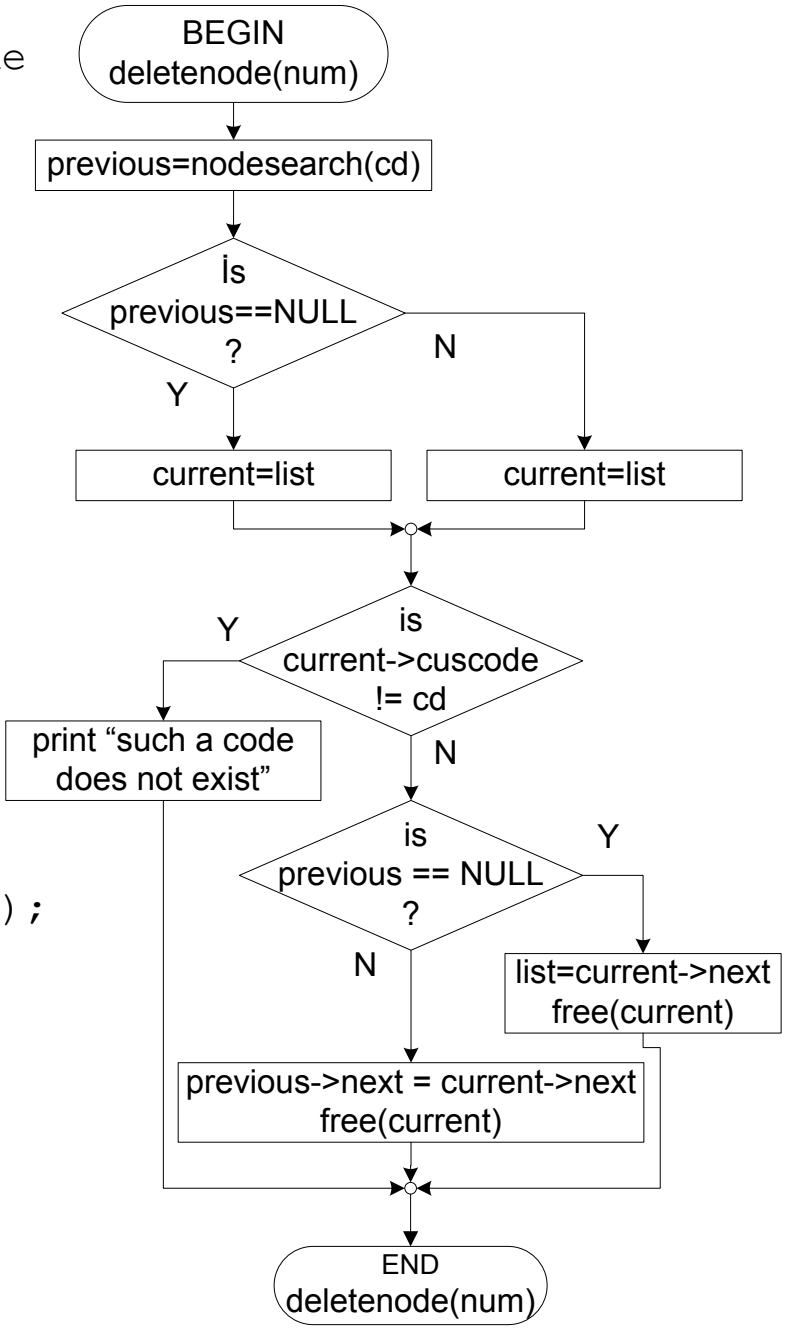



```
void display()
{
    node *currentnode;
    currentnode=list;
    printf("\nLinked list is:");
    printf("\n============= ");
    printf("\nCustomer code     Customer name\n");
    while(currentnode)
    {
    printf("\n%d %s",  currentnode->custcode,
                       currentnode->custname);
    currentnode=currentnode->next;

      if((currentnode->custcode)==-1)
      break;
    }
}
```

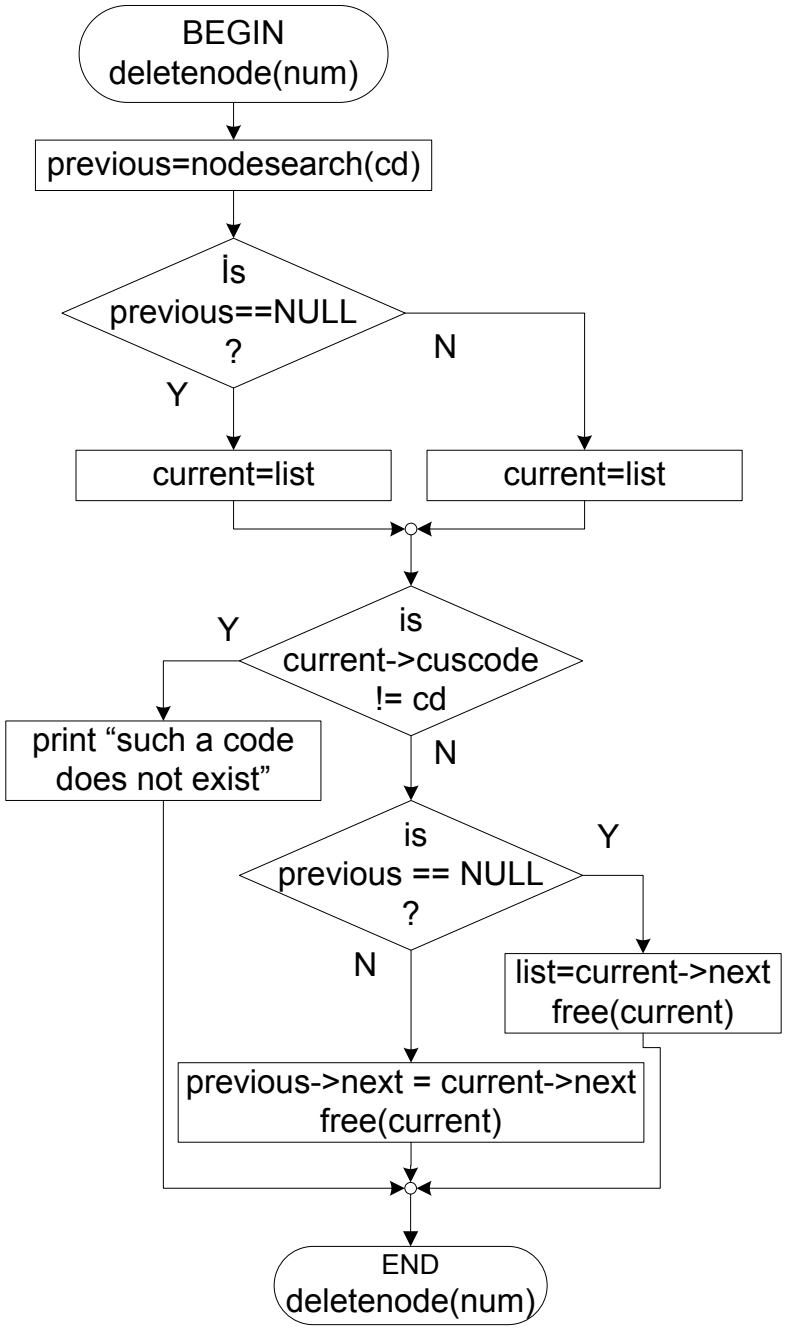


Algorithm to delete a node from a linked list

```
Step 1: Declare previous & current pointers as node
Step 2: Call nodesearch(cd) to find our if that
        customer code (cd) is previously used
        and assign the returned value 1 or 0
        to previous
Step 3: If the cd is not used before,
        i.e. previous == NULL,
        then assign the list pointer to current,
        otherwise assign previous->next to current
           if(previous==NULL)
              current=list;
           else
              current=previous->next;
Step 4: If the current->customer code != cd,
        print a message saying
        "Such a Code Does Not Exist".
        else if previous == NULL,
        then assign current->next to list,
        else
        assign current->next to previous->next.
           if((current->custcode)!=cd)
           {
              printf("Such a code doesn't exist!!");
           }
           else if(previous==NULL)
           {
              list=current->next;
           }
           else
           {
              previous->next=current->next;
              free(current);
  }
     }
```



```
void nodedelete(int cd)
{
  node *previous, *current;

  previous=nodesearch(cd);

  if(previous==NULL)
    current=list;
  else
    current=previous->next;

  if((current->custcode)!=cd)
  {
    printf("Such a code doesn't exist!");
    getch();
  }
  else if(previous==NULL)
  {
    list=current->next;
    free(current);
  }
  else
  {
    previous->next=current->next;
    free(current);
  }
}
```



```
// C PROGRAM TO CREATE DISPLAY INSERT DELETE A NODE IN A LINKED LIST
// This program is the collection of the above functions to form a complete and
// operating program helping students learn how to create a node, how to insert a node
// in an existing linked list, how the display the nodes and how to delete a node
// During preparation of these notes, the materials in the following web page are
// benefited from
// http://hubpages.com/hub/C-program-code-for-linked-list-manipulations
//
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>

struct linklist
{
  int custcode;
  char custname[30];
  struct linklist *next;
};
typedef struct linklist node;

node *list;
node *nodesearch(int);

void main()
{
  int n, cd;
  node *new, *temp;
  void nodecreate(node *);
  void insertnode(node *);
  void nodedelete(int);
  void display();

  temp=(node *)malloc(sizeof(node));
  list=temp;

  printf("\nLinked list creation");
  printf("\n====================");

  printf("\nEnter customer codes in ascending order\n\t\t(-1 to end):\n\n");
```

```
  nodecreate(temp);
  display();

  n=1;
  while(n!=3)
  {
    printf("\nLinked list manipulations\n");
    printf("-------------------------\n");

    printf("1. Insertion\n");
    printf("2. Deletion\n");
    printf("3. Exit\n");
    printf("Enter your choice:");
    scanf("%d", &n);

    switch(n)
    {
      case 1: new=(node *)malloc(sizeof(node));

          insertnode(new);
          display();

      break;

      case 2:printf("\n\t\tEnter the customercode to delete:");
        scanf("%d", &cd);
        nodedelete(cd);
        display();

      break;

    case 3: exit(1);

    }
  }
getch();
}

void nodecreate(node * ptr)
{
  printf("\n\t\tEnter code and press enter:");
```

```
  scanf("%d", &ptr->custcode);

  if((ptr->custcode)!=-1)
  {
    printf("\n\t\tEnter customer name:");
    scanf("%s", ptr->custname);

    ptr->next=(node *)malloc(sizeof(node));
    nodecreate(ptr->next);
  }
  else
    ptr->next=NULL;
}

void insertnode(node * new)
{
  int cd;
  char *cn;
  node *prev, *current;

  printf("\n\t\tEnter the customercode to insert:");
  scanf("%d", &cd);

  prev=nodesearch(cd);

  if(prev==NULL)
    current=list;
  else
    current=prev->next;

  if((current->custcode)==cd)
  {
    printf("\n\n\n\t\tCustomer code already exists!!");
    getch();
  }
  else
  {
    printf("\n\t\tEnter customername:");
    scanf("%s", cn);
    new->custcode=cd;
    strcpy(new->custname,cn);
```

```
  if(current==list)
  {
    new->next=current;
    list=new;
  }
  else
  {
    prev->next=new;
    new->next=current;
  }
 }
}

void nodedelete(int cd)
{
  node *prev, *current;

  prev=nodesearch(cd);

  if(prev==NULL)
    current=list;
  else
    current=prev->next;

  if((current->custcode)!=cd)
  {
    printf("\n\n\n\t\tSuch a code doesn't exist!!");
    getch();
  }
  else if(prev==NULL)
  {
    list=current->next;
    free(current);
  }
  else
  {
    prev->next=current->next;
    free(current);
  }
}
```

```
node * nodesearch(int cd)
{
  node *prev, *current;

  current=list;
  prev=NULL;

  while(cd>current->custcode)
  {
    if(current->next==NULL)
      break;
    else
    {
      prev=current;
      current=current->next;
    }
  }
return(prev);
}

void display()
{
  node *current;
  current=list;

  printf("\nLinked list is:");
  printf("\n============== ");
  printf("\n\nCustomer code\t\tCustomer name\n");

while(current)
  {
  printf("\n\n\t\t%d\t\t\t%s", current->custcode,current->custname);

  current=current->next;

  if((current->custcode)==-1)
    break;
  }
getch();
}
```