## Algorithm to delete middle node of Singly Linked List

**Algorithm to delete middle node of Singly Linked List**

**%%Input** : ***head*** node of the linked list

***n*** node to be deleted

**Begin:**

**If** (*head* == **NULL**) then

write ('List is already empty')

**End if**

**Else** then

*toDelete* ← *head*

*prevNode* ← *head*

**For** *i*←2 to *n* do

*prevNode* ← *toDelete*

*toDelete* ← *toDelete.next*

**If** (*toDelete* == **NULL**) then

**break**

**End if**

**End for**

**If** (*toDelete* != **NULL**) then

**If** (*toDelete* == *head*) then

*head* ← *head.next*

**End if**

*prevNode.next* ← *toDelete.next*

*toDelete.next* ← **NULL**

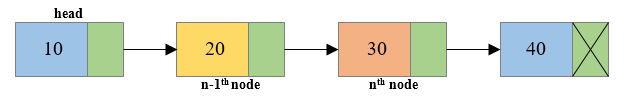
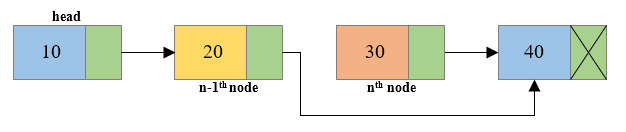
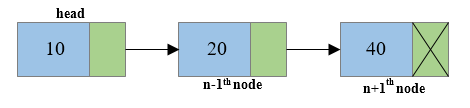
**unalloc** (*toDelete*)

**End if**

**End else**

**End**

**Steps to delete middle node of Singly Linked List**

1. Traverse to the nth node of the singly linked list and also keep reference of n-1th node in some temp variable say prevnode.
2. Reconnect the n-1th node with the n+1th node i.e. prevNode->next = toDelete->next (Where prevNode is n-1th node and toDelete node is the nth node and toDelete->next is the n+1th node).
3. Free the memory occupied by the nth node i.e. toDelete node.

**Program to delete middle node of Singly Linked List**

/\*\*

\* C program to delete middle node of Singly Linked List

\*/

#include <stdio.h>

#include <stdlib.h>

/\* Structure of a node \*/

struct node {

int data; // Data

struct node \*next; // Address

} \*head;

/\* Functions used in program \*/

void createList(int n);

void deleteMiddleNode(int position);

void displayList();

int main()

{

int n, position;

/\*

\* Create a singly linked list of n nodes

\*/

printf("Enter the total number of nodes: ");

scanf("%d", &n);

createList(n);

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

displayList();

printf("\nEnter the node position you want to delete: ");

scanf("%d", &position);

/\* Delete middle node from list \*/

deleteMiddleNode(position);

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

displayList();

return 0;

}

/\*

\* Create a list of n nodes

\*/

void createList(int n)

{

struct node \*newNode, \*temp;

int data, i;

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

/\*

\* If unable to allocate memory for head node

\*/

if(head == NULL)

{

printf("Unable to allocate memory.");

}

else

{

/\*

\* Read data of node from the user

\*/

printf("Enter the data of node 1: ");

scanf("%d", &data);

head->data = data; // Link the data field with data

head->next = NULL; // Link the address field to NULL

temp = head;

/\*

\* Create n nodes and adds to linked list

\*/

for(i=2; i<=n; i++)

{

newNode = (struct node \*)malloc(sizeof(struct node));

/\* If memory is not allocated for newNode \*/

if(newNode == NULL)

{

printf("Unable to allocate memory.");

break;

}

else

{

printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newNode->data = data; // Link the data field of newNode with data

newNode->next = NULL; // Link the address field of newNode with NULL

temp->next = newNode; // Link previous node i.e. temp to the newNode

temp = temp->next;

}

}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");

}

}

/\*

\* Delete middle node of the linked list

\*/

void deleteMiddleNode(int position)

{

int i;

struct node \*toDelete, \*prevNode;

if(head == NULL)

{

printf("List is already empty.");

}

else

{

toDelete = head;

prevNode = head;

for(i=2; i<=position; i++)

{

prevNode = toDelete;

toDelete = toDelete->next;

if(toDelete == NULL)

break;

}

if(toDelete != NULL)

{

if(toDelete == head)

head = head->next;

prevNode->next = toDelete->next;

toDelete->next = NULL;

/\* Delete nth node \*/

free(toDelete);

printf("SUCCESSFULLY DELETED NODE FROM MIDDLE OF LIST\n");

}

else

{

printf("Invalid position unable to delete.");

}

}

}

/\*

\* Display entire list

\*/

void displayList()

{

struct node \*temp;

/\*

\* If the list is empty i.e. head = NULL

\*/

if(head == NULL)

{

printf("List is empty.");

}

else

{

temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data); // Print the data of current node

temp = temp->next; // Move to next node

}

}

}

OUTPUT:

Output

Enter the total number of nodes: 4

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

SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list

Data = 10

Data = 20

Data = 30

Data = 40

Enter the node position you want to delete: 3

SUCCESSFULLY DELETED NODE FROM MIDDLE OF LIST

Data in the list

Data = 10

Data = 20

Data = 40