Assignments -01

1. Implement a singly connected linear linked list in C. Your program should typically implement insert and delete at all possible locations with proper check(s) as applicable. Include a display function as well and use it to show the content of your list after every operation. Include calls to insert and delete from the main.
2. Implement a singly connected circular linked list in C. Your program should typically implement insert and delete at all possible locations with proper check(s) as applicable. Include a display function as well and use it to show the content of your list after every operation. Include calls to insert and delete from the main.
3. Implement a doubly connected linear linked list in C. Your program should typically implement insert and delete at all possible locations with proper check(s) as applicable. Include a display function as well and use it to show the content of your list after every operation. Include calls to insert and delete from the main.
4. Implement a doubly connected circular linked list in C. Your program should typically implement insert and delete at all possible locations with proper check(s) as applicable. Include a display function as well and use it to show the content of your list after every operation. Include calls to insert and delete from the main. (For Bonus)

Solutions

Code (1):

SINGLY CONNECTED LINEAR LINKED LIST IN C

/\*This is the code for singly connected linear linked list

Done By

RAHUL SAHA

Roll: 001910701009

ETCE, UG2 ; Subject : DSA Lab \*/

*#include<stdio.h>*

*#include<stdlib.h>*

*struct Node{*

*int data;*

*struct Node\* next;*

*};*

*struct Node\* head;*

*void Print\_List();*

*void Insert\_At\_Beginning()*

*{*

*int val;*

*printf("\nEnter the value to be inserted: ");*

*scanf("%d",&val);*

*struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));*

*new\_node->data = val;*

*new\_node->next = head;*

*head = new\_node;*

*Print\_List();*

*}*

*void Insert\_At\_End()*

*{*

*int val;*

*printf("Enter the value to be inserted: ");*

*scanf("%d",&val);*

*struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));*

*new\_node->data = val;*

*new\_node->next = NULL;*

*if(head==NULL){*

*head = new\_node;*

*Print\_List();*

*}*

*else{*

*struct Node\* previ\_node = head;*

*while(previ\_node->next!=NULL)*

*previ\_node=previ\_node->next;*

*previ\_node->next = new\_node;*

*}*

*Print\_List();*

*}*

*int count()*

*{*

*int cnt=0;*

*struct Node\* current = head;*

*while(current!=NULL)*

*{*

*cnt++;*

*current=current->next;*

*}*

*return cnt;*

*}*

*void Insert\_At\_AnyPosition()*

*{*

*int val,pos;*

*printf("Enter the value to be inserted: ");*

*scanf("%d",&val);*

*printf("Enter the position of insertion: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(pos<=0 || pos>=c+2)*

*{*

*printf("\nUser have entered Invalid Position");*

*return;*

*}*

*struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));*

*new\_node->data = val;*

*new\_node->next=NULL;*

*if(pos==1)*

*{*

*new\_node->next = head;*

*head=new\_node;*

*Print\_List();*

*}*

*else*

*{*

*struct Node\* previ\_node=head;*

*for(int i=0;i<pos-2;i++)*

*previ\_node = previ\_node->next;*

*new\_node->next = previ\_node->next;*

*previ\_node->next = new\_node;*

*Print\_List();*

*}*

*}*

*void Delete()*

*{*

*int pos;*

*printf("Enter the position to be deleted: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(c==0)*

*{*

*printf("---------UNDERFLOW DETECTED--------\n");*

*printf("---------NO ELEMENT FOUND FOR DELETION---------\n");*

*return;*

*}*

*if(pos<=0 || pos>c)*

*{*

*printf("\nUser have entered Invalid Position");*

*return;*

*}*

*struct Node\* del\_node = head;*

*if(pos==1)*

*{*

*head = del\_node->next;*

*free(del\_node);*

*if(c==1)*

*{*

*printf(" List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*Print\_List();*

*return;*

*}*

*int i;*

*for(i=0;i<pos-2;i++)*

*del\_node = del\_node->next;*

*struct Node\* del1\_node = del\_node->next;*

*del\_node->next =del1\_node->next;*

*free(del1\_node);*

*if(c==1)*

*{*

*printf("List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*Print\_List();*

*}*

*void Print\_List()*

*{*

*struct Node\* print\_node = head;*

*printf("\n The list is: ");*

*while(print\_node!=NULL)*

*{*

*printf("%d ",print\_node->data);*

*print\_node = print\_node->next;*

*}*

*}*

*int main()*

*{*

*head = NULL;*

*int choose,num=1;*

*while(num==1)*

*{*

*printf("Enter 1 for inserting an element at the beginning of the list \nEnter 2 for inserting at any position in the list \nEnter 3 for inserting at end of the list \nEnter 4 for deleting an element from any position in the list \nEnter 5 for viewing the final list");*

*printf("\nEnter your choice: ");*

*scanf("%d",&choose);*

*if(choose == 1)*

*Insert\_At\_Beginning();*

*else if(choose == 2)*

*Insert\_At\_AnyPosition();*

*else if(choose == 3)*

*Insert\_At\_End();*

*else if(choose == 4)*

*Delete();*

*else if(choose == 5)*

*Print\_List();*

*else*

*printf("Enter valid choice\n");*

*printf("\nEnter 1 to continue and 0 to exit");*

*printf("\nDo you want to continue?: ");*

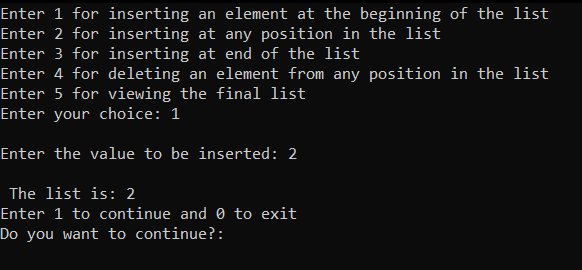
*scanf("%d",&num);*

*}*

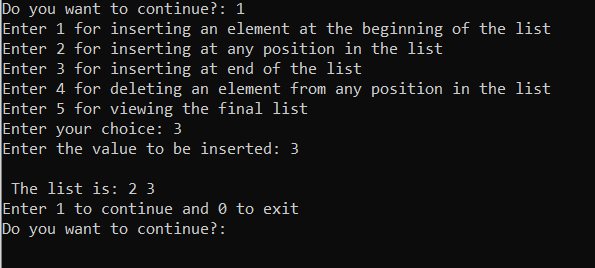
*}*

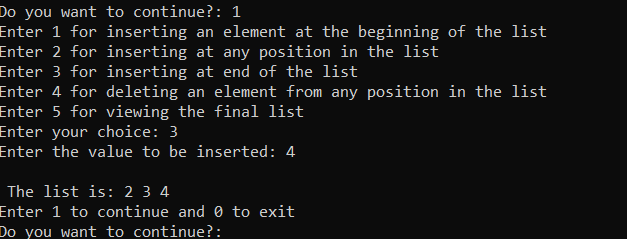
**Outputs**

Insertion At Beginning:

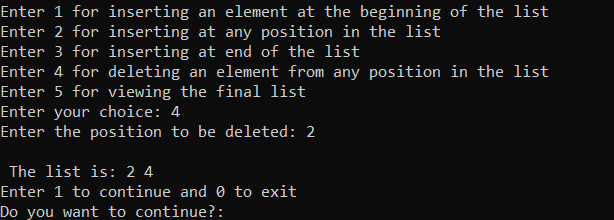


Insert At End

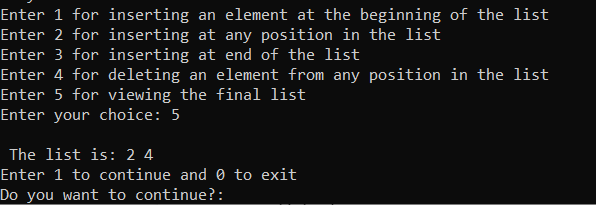




Delete at nth position



Viewing the final list



**Code (2):**

*/\*This is the code for singly connected circular linked list*

*Done By*

*RAHUL SAHA*

*Roll: 001910701009*

*ETCE, UG2 ; Subject : DSA Lab \*/*

*#include<stdio.h>*

*#include<stdlib.h>*

*struct Node{*

*int data;*

*struct Node\* next;*

*};*

*struct Node\* head;*

*struct Node\* create\_node(x)*

*{*

*struct Node\* temp = (struct Node\*)malloc(sizeof(struct Node));*

*temp->data = x;*

*temp->next = temp;*

*return temp;*

*}*

*int count()*

*{*

*struct Node\* current = head;*

*int cnt=0;*

*if(head!=NULL)*

*{*

*do*

*{*

*cnt++;*

*current = current->next;*

*}while(current!=head);*

*}*

*return cnt;*

*}*

*void Print\_List()*

*{*

*struct Node\* temp = head;*

*printf("\nThe list is: ");*

*if(head!=NULL)*

*{*

*do*

*{*

*printf("\n Value: %d It's address : %u Next Node Address: %u ",temp->data,temp,temp->next);*

*temp = temp->next;*

*}while(temp!=head);*

*}*

*}*

*void Insert\_At\_Beginning()*

*{*

*int x;*

*printf("\nEnter the value: ");*

*scanf("%d",&x);*

*struct Node\* new\_val = create\_node(x);*

*if(head == NULL)*

*{*

*head = new\_val;*

*new\_val->next = head;*

*Print\_List();*

*return;*

*}*

*else*

*{*

*struct Node\* prev\_node = head;*

*while(prev\_node->next!=head)*

*prev\_node = prev\_node->next;*

*new\_val->next = head;*

*prev\_node->next = new\_val;*

*head = new\_val;*

*Print\_List();*

*}*

*}*

*void Insert\_At\_AnyPosition()*

*{*

*int val,pos;*

*printf("Enter the value: ");*

*scanf("%d",&val);*

*printf("Enter the position: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(pos<=0 || pos>=c+2)*

*{*

*printf("\nError! Invalid position");*

*return;*

*}*

*struct Node\* new\_node = create\_node(val);*

*if(pos==1)*

*{*

*Insert\_At\_Beginning(pos);*

*return;*

*}*

*struct Node\* prev\_val = head;*

*for(int i=0;i<pos-2;i++)*

*prev\_val = prev\_val->next;*

*new\_node->next = prev\_val->next;*

*prev\_val->next = new\_node;*

*Print\_List();*

*}*

*void Insert\_At\_End() //The function is to append the new values in the list*

*{*

*int val;*

*printf("Enter the value: ");*

*scanf("%d",&val);*

*struct Node\* new\_node = create\_node(val);*

*if(head==NULL)*

*{*

*head = new\_node;*

*Print\_List();*

*return;*

*}*

*else*

*{*

*struct Node\* prev\_node = head;*

*while(prev\_node->next!=head)*

*prev\_node = prev\_node->next;*

*new\_node->next = head;*

*prev\_node->next = new\_node;*

*Print\_List();*

*}*

*}*

*void Delete()*

*{*

*int pos;*

*printf("Enter the position: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(c==0)*

*{*

*printf("---------UNDERFLOW DETECTED--------\n");*

*printf("---------NO ELEMENT FOUND FOR DELETION---------\n");*

*return;*

*}*

*if(pos<=0 || pos >= c+2) //Position 0 means no element to delete, the pointer moves before the 1st element and if greater than c +2 then pointer moves after the last element*

*{*

*printf("\nUser have entered Invalid Position");*

*return;*

*}*

*struct Node\* del\_node = head;*

*if(pos==1)*

*{*

*while(del\_node->next!=head)*

*del\_node= del\_node->next;*

*del\_node->next = head->next;*

*free(head);*

*head = del\_node->next;*

*if(c==1)*

*{*

*printf("List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*Print\_List();*

*return;*

*}*

*else*

*{*

*for(int i=0;i<pos-2;i++)*

*del\_node = del\_node->next;*

*struct Node\* del1\_node=del\_node->next;*

*del\_node->next = del1\_node->next;*

*free(del1\_node);*

*if(c==1)*

*{*

*printf("List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*Print\_List();*

*}*

*}*

*int main()*

*{*

*head = NULL;*

*int choose,num=1;*

*while(num==1)*

*{*

*printf("Enter 1 for inserting an element at the beginning of the list\nEnter 2 for inserting at any position in the list\nEnter 3 for inserting at end of the list\nEnter 4 for deleting an element from the list\nEnter 5 to view the latest list");*

*printf("\nEnter your choice: ");*

*scanf("%d",&choose);*

*if(choose == 1)*

*Insert\_At\_Beginning();*

*else if(choose == 2)*

*Insert\_At\_AnyPosition();*

*else if(choose == 3)*

*Insert\_At\_End();*

*else if(choose == 4)*

*Delete();*

*else if(choose == 5)*

*Print\_List();*

*else*

*printf("Enter valid choice\n");*

*printf("\nEnter 1 to continue and 0 to exit");*

*printf("\nDo you want to continue?: ");*

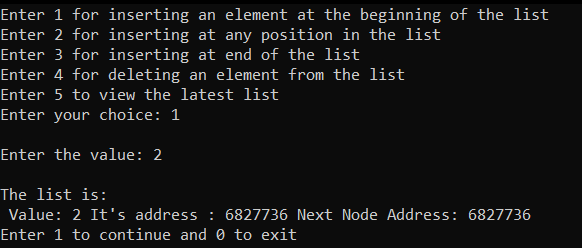
*scanf("%d",&num);*

*}*

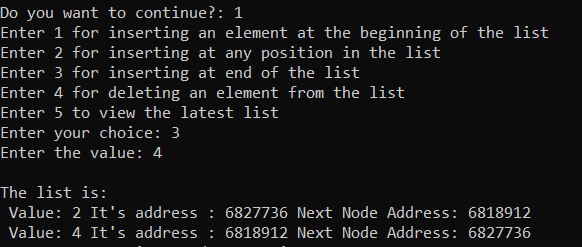
*}*

**Outputs:**

Insert Node At The Beginning

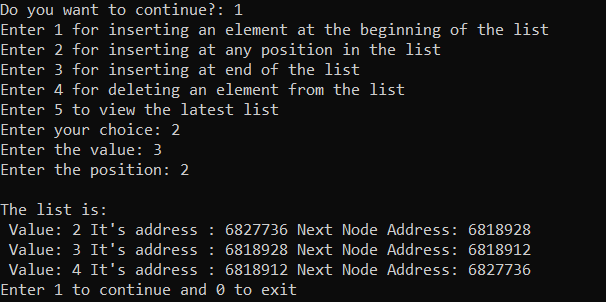


Insert At The End

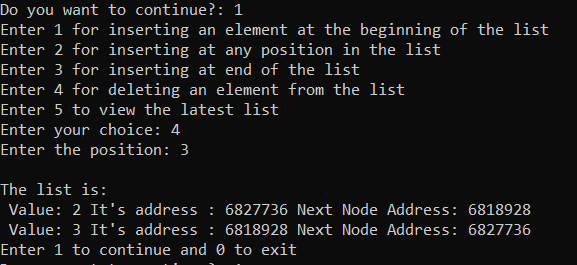


Clearly visible that the address of the next node in the last element is basically the address of the first node, hence the list is circular.

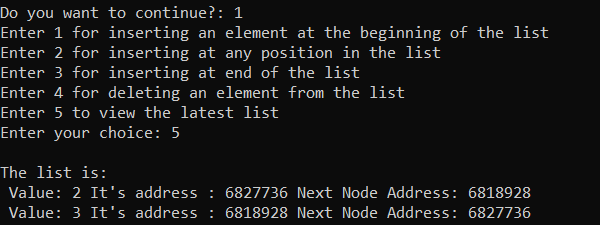
Insert At Any Position



Delete at nth position



View the final list



**Code (3):**

DOUBLY CONNECTED LINEAR LINKED LIST

*/\*This is the code for doubly connected linear linked list*

*Done By*

*RAHUL SAHA*

*Roll: 001910701009*

*ETCE, UG2 ; Subject : DSA Lab \*/*

*#include <stdio.h>*

*#include <stdlib.h>*

*struct Node{*

*int data;*

*struct Node\* next;*

*struct Node\* prev;*

*};*

*struct Node\* head;*

*struct Node\* create\_node(int val)*

*{*

*struct Node\* temp = (struct Node\*)malloc(sizeof(struct Node));*

*temp->data = val;*

*temp->next = NULL;*

*temp->prev = NULL;*

*return temp;*

*}*

*int count();*

*void Print\_List();*

*void Insert\_At\_Beginning()*

*{*

*int val;*

*printf("Enter the value to be inserted: ");*

*scanf("%d",&val);*

*struct Node\* new\_node = create\_node(val);*

*if(head == NULL)*

*{*

*head = new\_node;*

*Print\_List();*

*return;*

*}*

*head->prev = new\_node;*

*new\_node->next = head;*

*head = new\_node;*

*Print\_List();*

*}*

*void Insert\_At\_AnyPosition()*

*{*

*int val,pos;*

*printf("Enter the value: ");*

*scanf("%d",&val);*

*printf("Enter the position: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(pos<=0 || pos>=c+2)*

*{*

*printf("\nUser have entered Invalid Position");*

*return;*

*}*

*struct Node\* new\_node = create\_node(val);*

*if(pos==1)*

*{*

*new\_node->next = head;*

*head = new\_node;*

*Print\_List();*

*return;*

*}*

*struct Node\* prev\_node = head;*

*for(int i=0;i<pos-2;i++)*

*prev\_node = prev\_node->next;*

*new\_node->next = prev\_node->next;*

*prev\_node->next = new\_node;*

*new\_node->prev = prev\_node;*

*if(new\_node->next!=NULL)*

*new\_node->next->prev = new\_node;*

*Print\_List();*

*return;*

*}*

*void Insert\_At\_End()*

*{*

*int val;*

*printf("Enter the value to be inserted: ");*

*scanf("%d",&val);*

*struct Node\* new\_node = create\_node(val);*

*if(head == NULL)*

*{*

*head = new\_node;*

*Print\_List();*

*return;*

*}*

*struct Node\* prev\_node = head;*

*while(prev\_node->next!=NULL)*

*prev\_node = prev\_node->next;*

*prev\_node->next = new\_node;*

*new\_node->prev = prev\_node;*

*Print\_List();*

*return;*

*}*

*void Delete()*

*{*

*int pos;*

*printf("Enter the position to be deleted: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(c==0)*

*{*

*printf("---------UNDERFLOW DETECTED--------\n");*

*printf("---------NO ELEMENT FOUND FOR DELETION---------\n");*

*return;*

*}*

*if(pos<=0 || pos>c)*

*{*

*printf("%d\n",c);*

*printf("\nUser have entered Invalid Position");*

*return;*

*}*

*struct Node\* del\_node = head;*

*if(pos==1)*

*{*

*head = del\_node->next;*

*free(del\_node);*

*if(c==1)*

*{*

*printf("List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*Print\_List();*

*return;*

*}*

*for(int j=0;j<pos-2;j++)*

*del\_node = del\_node->next;*

*struct Node\* del\_node1 = del\_node->next;*

*del\_node->next = del\_node1->next;*

*if(del\_node1->next!=NULL)*

*del\_node1->next->prev = del\_node;*

*free(del\_node1);*

*if(c==1)*

*{*

*printf("List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*Print\_List();*

*}*

*void Print\_List()*

*{*

*struct Node\* temp\_print = head;*

*printf("\nThe list is: ");*

*while(temp\_print!=NULL)*

*{*

*printf("%d ",temp\_print->data);*

*temp\_print = temp\_print->next;*

*}*

*printf("\n");*

*}*

*int count()*

*{*

*struct Node\* current = head;*

*int cnt=0;*

*while(current!=NULL)*

*{*

*cnt++;*

*current = current->next;*

*}*

*return cnt;*

*}*

*int main()*

*{*

*head = NULL;*

*int choose,num=1;*

*while(num==1)*

*{*

*printf("Enter 1 for inserting an element at the beginning\nEnter 2 for inserting at any position\nEnter 3 for inserting at end\nEnter 4 for deleting an element\nEnter 5 for viewing latest list");*

*printf("\nEnter your choice: ");*

*scanf("%d",&choose);*

*if(choose == 1)*

*Insert\_At\_Beginning();*

*else if(choose == 2)*

*Insert\_At\_AnyPosition();*

*else if(choose == 3)*

*Insert\_At\_End();*

*else if(choose == 4)*

*Delete();*

*else if(choose == 5)*

*Print\_List();*

*else*

*printf("Enter valid choice\n");*

*printf("\nEnter 1 to continue and 0 to exit");*

*printf("\nDo you want to continue?: ");*

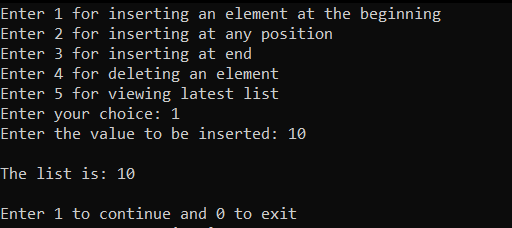
*scanf("%d",&num);*

*}*

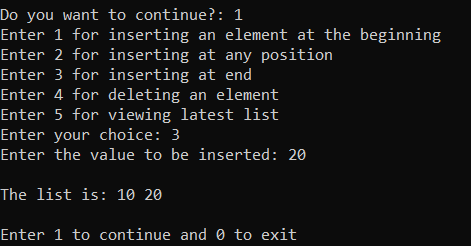
*}*

**Outputs:**

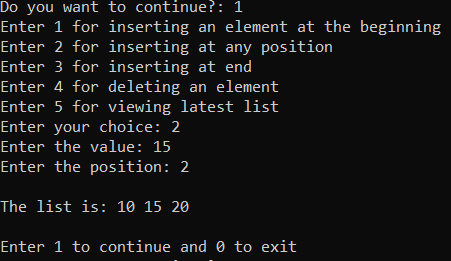
Insertion At Beginning



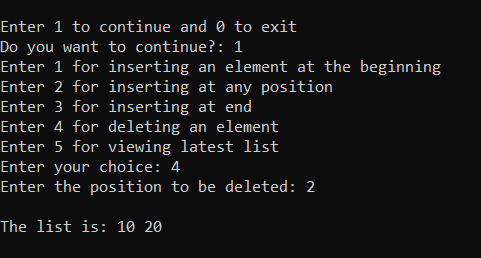
Insertion At Last



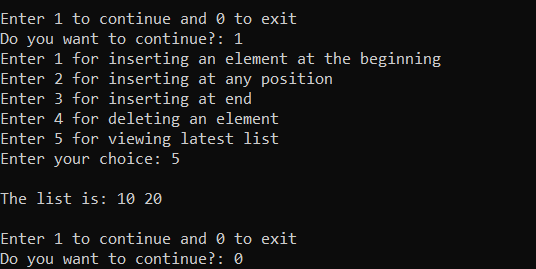
Insertion At nth position



Deletion At nth position



Final List



**Code (4):**

DOUBLY CONNECTED CIRCULAR LINKED LIST

*/\*This is the code for doubly connected circular linked list*

*Done By*

*RAHUL SAHA*

*Roll: 001910701009*

*ETCE, UG2 ; Subject : DSA Lab \*/*

*#include<stdio.h>*

*#include<stdlib.h>*

*struct Node{*

*int data;*

*struct Node\* next;*

*struct Node\* prev;*

*};*

*struct Node\* head;*

*struct Node\* create\_node(val)*

*{*

*struct Node\* temp = (struct Node\*)malloc(sizeof(struct Node));*

*temp->data = val;*

*temp->next = temp;*

*temp->prev = temp;*

*return temp;*

*}*

*void Print\_List();*

*int count();*

*void Insert\_At\_Beginning()*

*{*

*int val;*

*printf("Enter the value to be inserted: ");*

*scanf("%d",&val);*

*struct Node\* new\_val = create\_node(val);*

*if(head == NULL)*

*{*

*head = new\_val;*

*new\_val->next = head;*

*Print\_List();*

*return;*

*}*

*else*

*{*

*struct Node\* prev\_node = head;*

*while(prev\_node->next!=head)*

*prev\_node = prev\_node->next;*

*new\_val->next = head;*

*prev\_node->next = new\_val;*

*head = new\_val;*

*Print\_List();*

*}*

*}*

*void Insert\_At\_AnyPosition()*

*{*

*int val,pos;*

*printf("Enter the value to be inserted: ");*

*scanf("%d",&val);*

*printf("Enter the position of insertion: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(pos<=0 || pos>=c+2)*

*{*

*printf("\nUser have entered Invalid Position");*

*return;*

*}*

*struct Node\* new\_node = create\_node(val);*

*if(pos==1)*

*{*

*Insert\_At\_Beginning(val);*

*return;*

*}*

*struct Node\* prev\_node = head;*

*for(int i=0;i<pos-2;i++)*

*prev\_node = prev\_node->next;*

*new\_node->next = prev\_node->next;*

*prev\_node->next = new\_node;*

*new\_node->prev = prev\_node;*

*Print\_List();*

*return;*

*}*

*void Insert\_At\_End()*

*{*

*int val;*

*printf("Enter the value: ");*

*scanf("%d",&val);*

*struct Node\* new\_node = create\_node(val);*

*if(head == NULL)*

*{*

*head = new\_node;*

*Print\_List();*

*return;*

*}*

*struct Node\* prev\_node = head;*

*while(prev\_node->next!=head)*

*prev\_node = prev\_node->next;*

*new\_node->next = prev\_node->next;*

*prev\_node->next = new\_node;*

*new\_node->prev = prev\_node;*

*new\_node->next->prev = new\_node;*

*Print\_List();*

*return;*

*}*

*void Delete()*

*{*

*int pos;*

*printf("Enter the position of deletion: ");*

*scanf("%d",&pos);*

*int c = count();*

*if(c==0)*

*{*

*printf("---------UNDERFLOW DETECTED--------\n");*

*printf("---------NO ELEMENT FOUND FOR DELETION---------\n");*

*return;*

*}*

*if(pos<=0 || pos>c)*

*{*

*printf("\nUser have entered Invalid Position");*

*return;*

*}*

*struct Node\* del\_node = head;*

*if(pos==1)*

*{*

*while(del\_node->next!=head)*

*del\_node = del\_node->next;*

*del\_node->next = head->next;*

*head->next->prev = del\_node;*

*free(head);*

*if(c==1)*

*{*

*printf("List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*head = del\_node->next;*

*Print\_List();*

*return;*

*}*

*for(int j=0;j<pos-2;j++)*

*del\_node = del\_node->next;*

*struct Node\* del\_node1 = del\_node->next;*

*del\_node->next = del\_node1->next;*

*del\_node1->next->prev = del\_node;*

*free(del\_node1);*

*if(c==1)*

*{*

*printf("List Is Empty, No Element Found For Deletion\n");*

*return;*

*}*

*Print\_List();*

*}*

*void Print\_List()*

*{*

*struct Node\* temp\_node = head;*

*printf("\nThe list is: ");*

*if(head!=NULL)*

*{*

*do*

*{*

*printf(" %d ",temp\_node->data);*

*temp\_node = temp\_node->next;*

*}while(temp\_node!=head);*

*}*

*}*

*int count()*

*{*

*struct Node\* current = head;*

*int cnt=0;*

*if(head!=NULL)*

*{*

*do*

*{*

*cnt++;*

*current = current->next;*

*}while(current!=head);*

*}*

*return cnt;*

*}*

*int main()*

*{*

*head = NULL;*

*int choose,num=1;*

*while(num==1)*

*{*

*printf("Enter 1 for inserting an element at the beginning\nEnter 2 for inserting at any position\nEnter 3 for inserting at end\nEnter 4 for deleting an element\nEnter 5 for viewing the latest list");*

*printf("\nEnter your choice: ");*

*scanf("%d",&choose);*

*if(choose == 1)*

*Insert\_At\_Beginning();*

*else if(choose == 2)*

*Insert\_At\_AnyPosition();*

*else if(choose == 3)*

*Insert\_At\_End();*

*else if(choose == 4)*

*Delete();*

*else if(choose == 5)*

*Print\_List();*

*else*

*printf("Enter valid choice\n");*

*printf("\nEnter 1 to continue and 0 to exit");*

*printf("\nDo you want to continue?: ");*

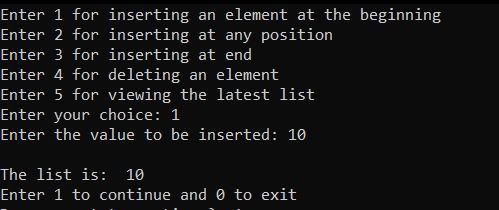
*scanf("%d",&num);*

*}*

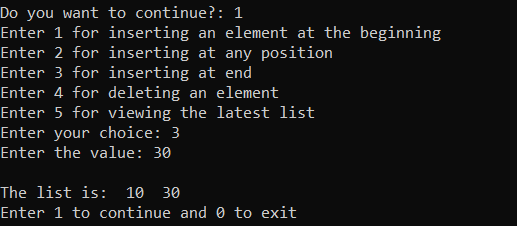
*}*

**Outputs:**

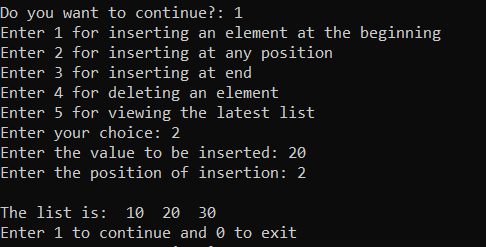
Insert At Beginning



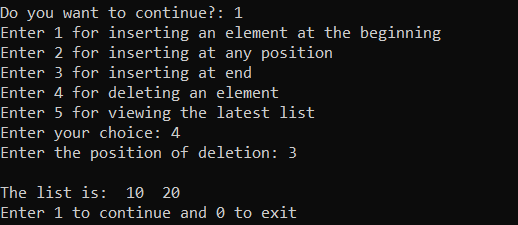
Insert At End



Insert At nth position



Delete at nth position



View the final list

