**CA3001 – Programming and Data Structures using C**

**Assignment 14 - 15.02.2021**

**SINGLY LINKED LIST OPERATIONS:**

**Q1A. Write a program to create a singly linked list of n nodes**

**Ans – C Program:**

#include <stdio.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head, \*tail = NULL;

void addNode(int data)

{

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

newNode->data = data;

newNode->next = NULL;

if(head == NULL)

{

head = newNode;

tail = newNode;

}

else

{

tail->next = newNode;

tail = newNode;

}

}

int countNodes()

{

int count = 0;

struct node \*current = head;

while(current != NULL)

{

count++;

current = current->next;

}

return count;

}

void display()

{

struct node \*current = head;

if(head == NULL)

{

printf("List is empty\n");

return;

}

printf("Nodes of singly linked list: \n");

while(current != NULL)

{

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

int main()

{

addNode(1);

addNode(2);

addNode(3);

addNode(4);

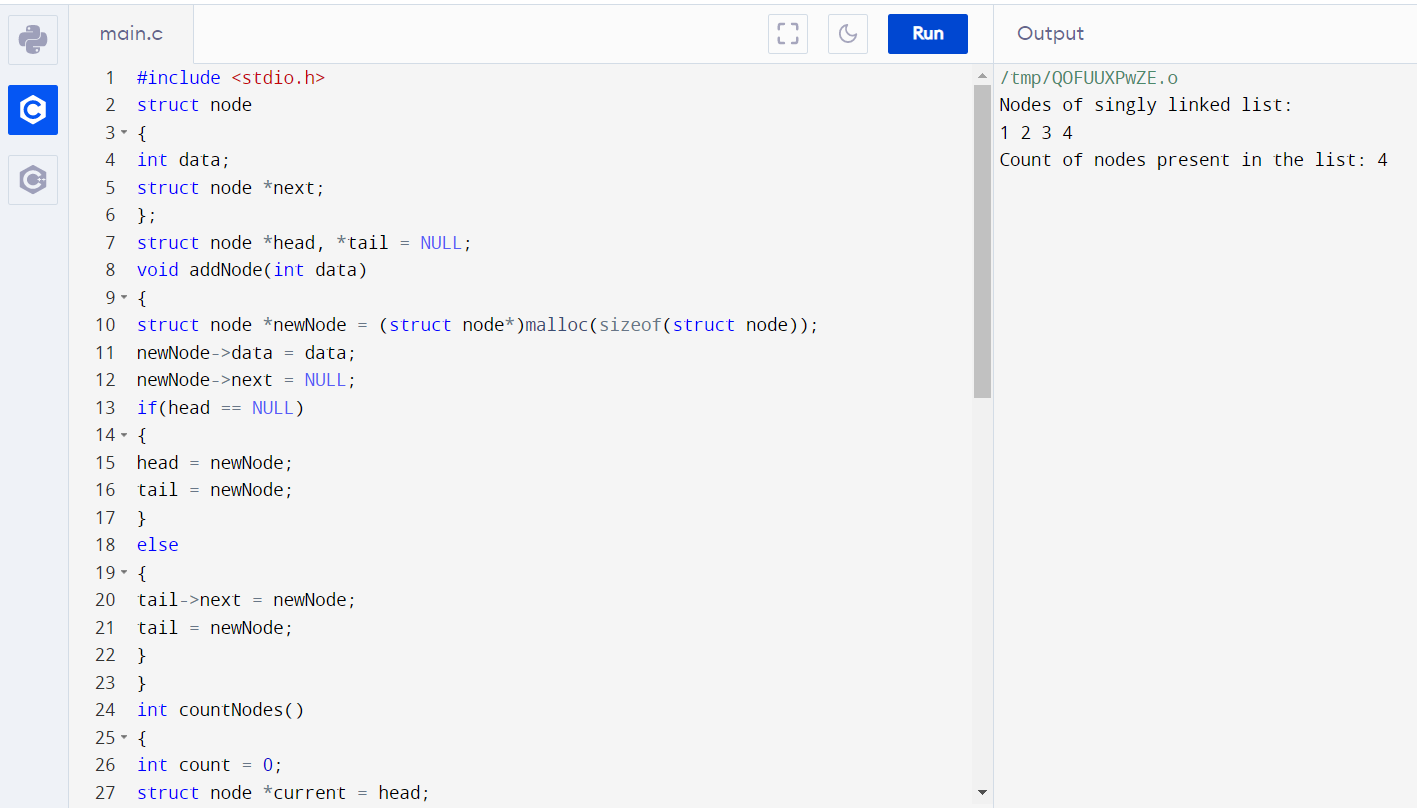
display();

printf("Count of nodes present in the list: %d", countNodes());

return 0;

}

**Output:**



**Q1B. Write a program to insert a new node at the beginning**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void beginsert(int);

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice,item;

do

{

printf("\nEnter the item which you want to insert?\n");

scanf("%d",&item);

beginsert(item);

printf("\nPress 0 to insert more ?\n");

scanf("%d",&choice);

}while(choice == 0);

}

void beginsert(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

}

else

{

ptr->data = item;

ptr->next = head;

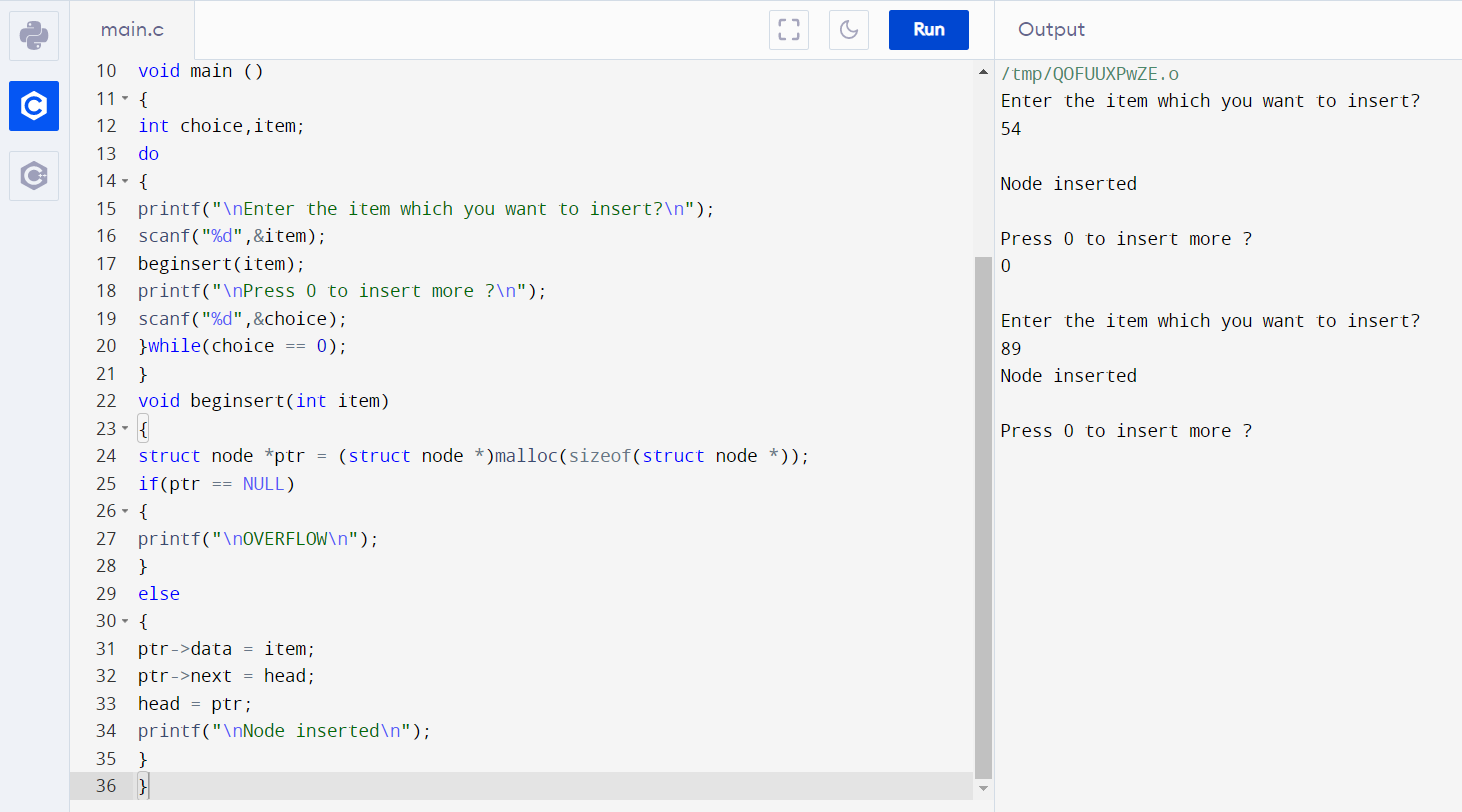
head = ptr;

printf("\nNode inserted\n");

}

}

**Output:**



**Q1C. Write a program to insert a new node at the end.**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void lastinsert(int);

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice,item;

do

{

printf("\nEnter the item which you want to insert?\n");

scanf("%d",&item);

lastinsert(item);

printf("\nPress 0 to insert more ?\n");

scanf("%d",&choice);

}while(choice == 0);

}

void lastinsert(int item)

{

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

struct node \*temp;

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

ptr->data = item;

if(head == NULL)

{

ptr -> next = NULL;

head = ptr;

printf("\nNode inserted");

}

else

{

temp = head;

while (temp -> next != NULL)

{

temp = temp -> next;

}

temp->next = ptr;

ptr->next = NULL;

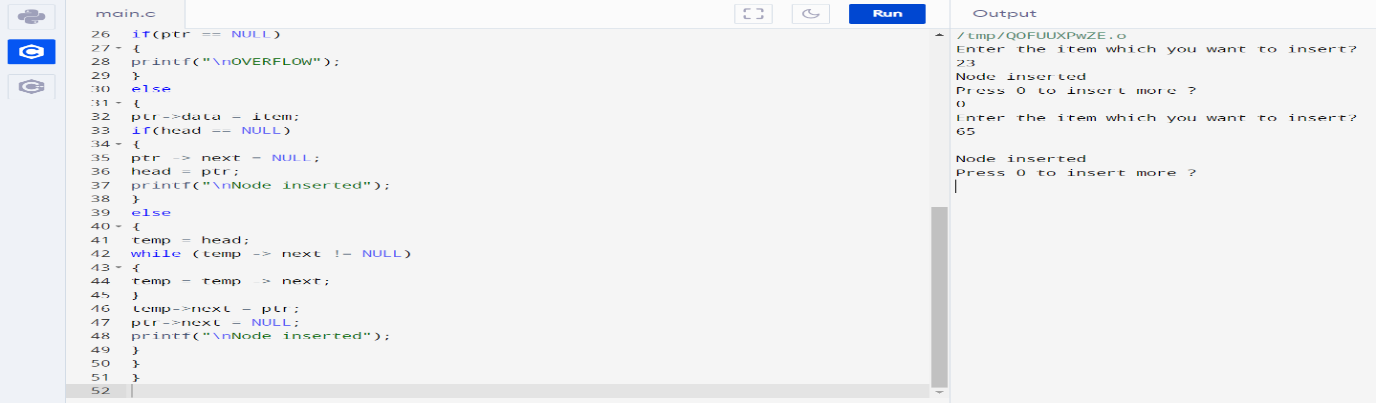
printf("\nNode inserted");

}

}

}

**Output:**



**Q1d. Write a program to insert a new node at any location.**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void randominsert(int);

void create(int);

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice,item,loc;

do

{

printf("\nEnter the item which you want to insert?\n");

scanf("%d",&item);

if(head == NULL)

{

create(item);

}

else

{

randominsert(item);

}

printf("\nPress 0 to insert more ?\n");

scanf("%d",&choice);

}while(choice == 0);

}

void create(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

}

else

{

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nNode inserted\n");

}

}

void randominsert(int item)

{

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

struct node \*temp;

int i,loc;

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("Enter the location");

scanf("%d",&loc);

ptr->data = item;

temp=head;

for(i=0;i<loc;i++)

{

temp = temp->next;

if(temp == NULL)

{

printf("\ncan't insert\n");

return;

}

}

ptr ->next = temp ->next;

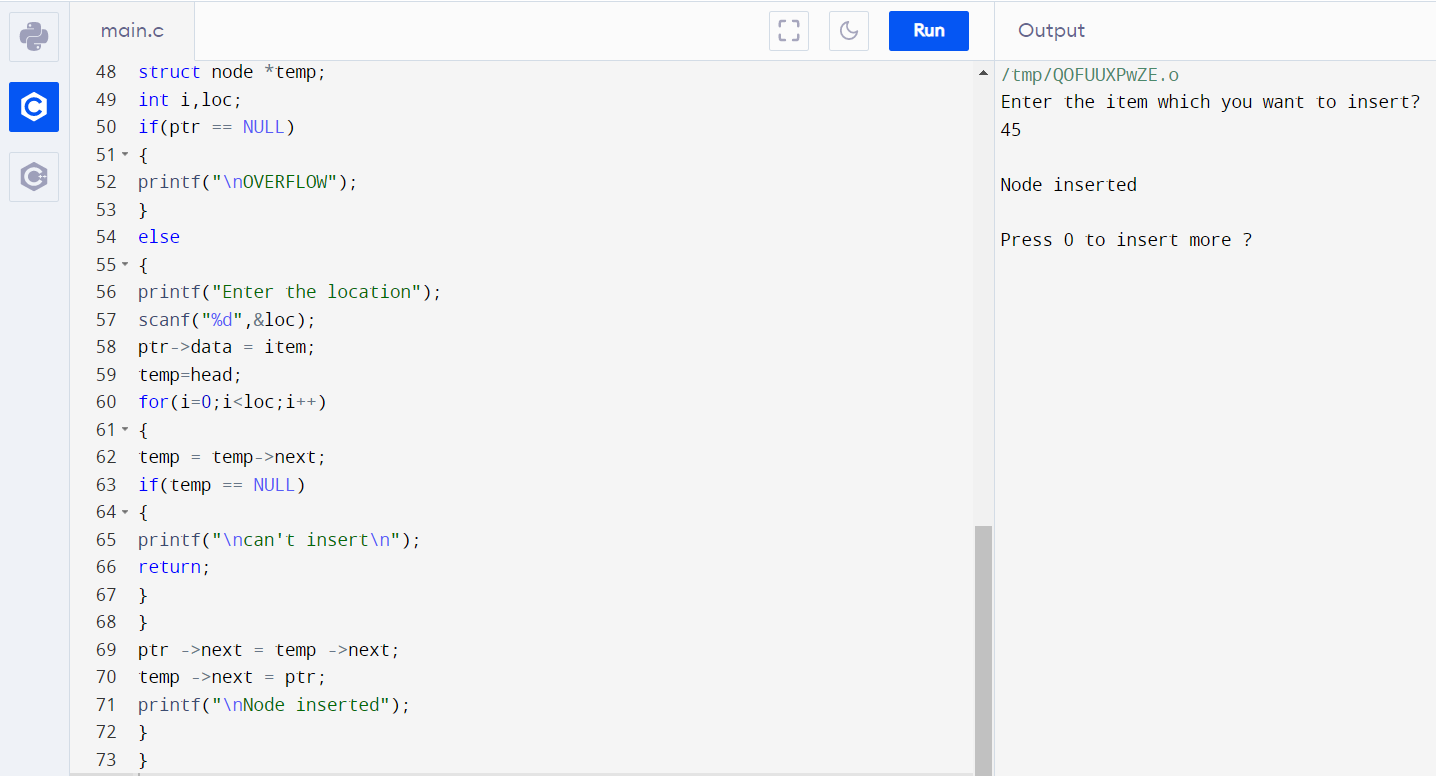
temp ->next = ptr;

printf("\nNode inserted");

}

}

**Output:**



**Q1E. Write a program to delete a node at the beginning**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void create(int);

void begdelete();

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice,item;

do

{

printf("\n1.Append List\n2.Delete node\n3.Exit\n4.Enter your choice?");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the item\n");

scanf("%d",&item);

create(item);

break;

case 2:

begdelete();

break;

case 3:

exit(0);

break;

default:

printf("\nPlease enter valid choice\n");

}

}while(choice != 3);

}

void create(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

}

else

{

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nNode inserted\n");

}

}

void begdelete()

{

struct node \*ptr;

if(head == NULL)

{

printf("\nList is empty");

}

else

{

ptr = head;

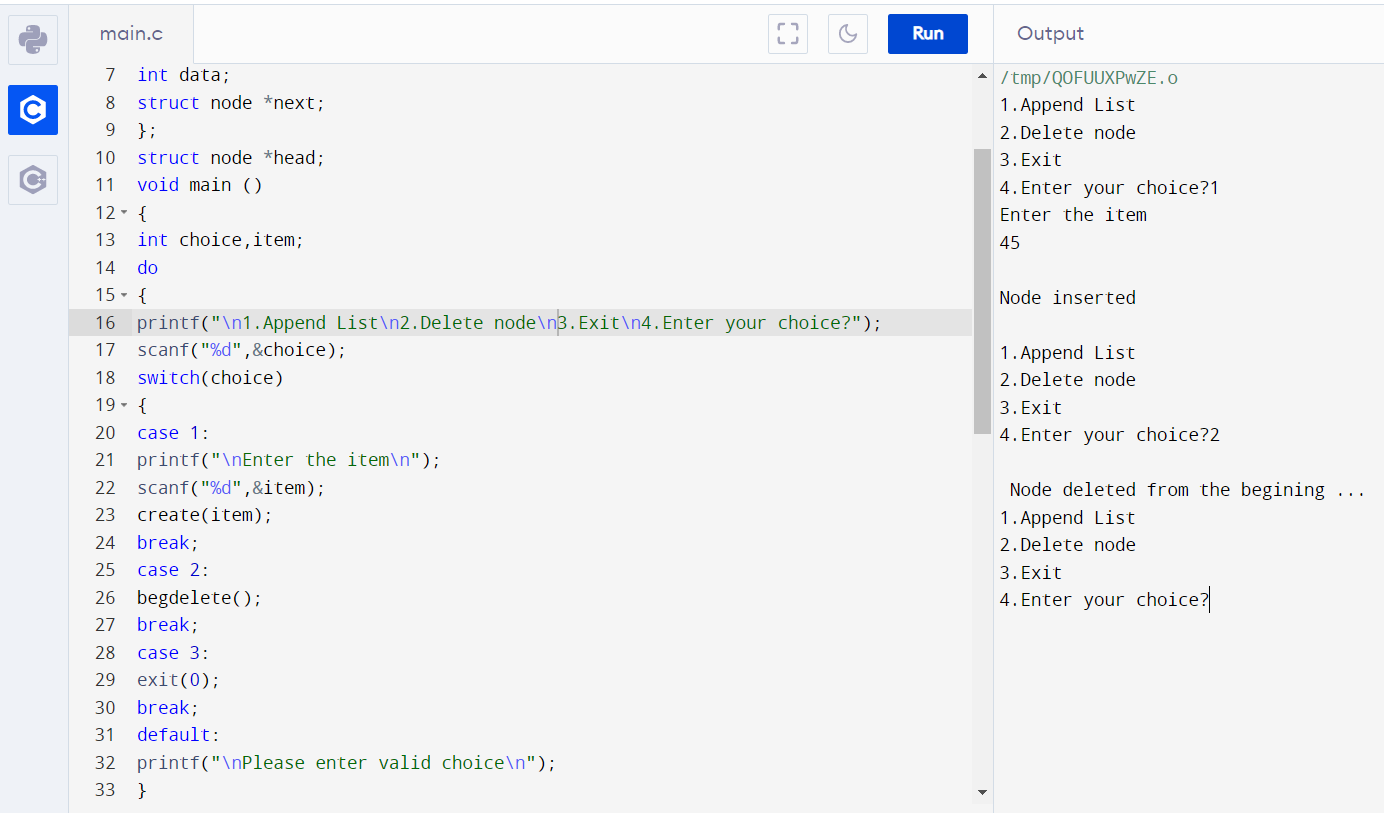
head = ptr->next;

free(ptr);

printf("\n Node deleted from the begining ...");

}}

**Output:**



**Q1F. Write a program to delete a node at the end**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void create(int);

void end\_delete();

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice,item;

do

{

printf("\n1.Append List\n2.Delete node\n3.Exit\n4.Enter your choice?");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the item\n");

scanf("%d",&item);

create(item);

break;

case 2:

end\_delete();

break;

case 3:

exit(0);

break;

default:

printf("\nPlease enter valid choice\n");

}

}while(choice != 3);

}

void create(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

}

else

{

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nNode inserted\n");

}

}

void end\_delete()

{

struct node \*ptr,\*ptr1;

if(head == NULL)

{

printf("\nlist is empty");

}

else if(head -> next == NULL)

{

head = NULL;

free(head);

printf("\nOnly node of the list deleted ...");

}

else

{

ptr = head;

while(ptr->next != NULL)

{

ptr1 = ptr;

ptr = ptr ->next;

}

ptr1->next = NULL;

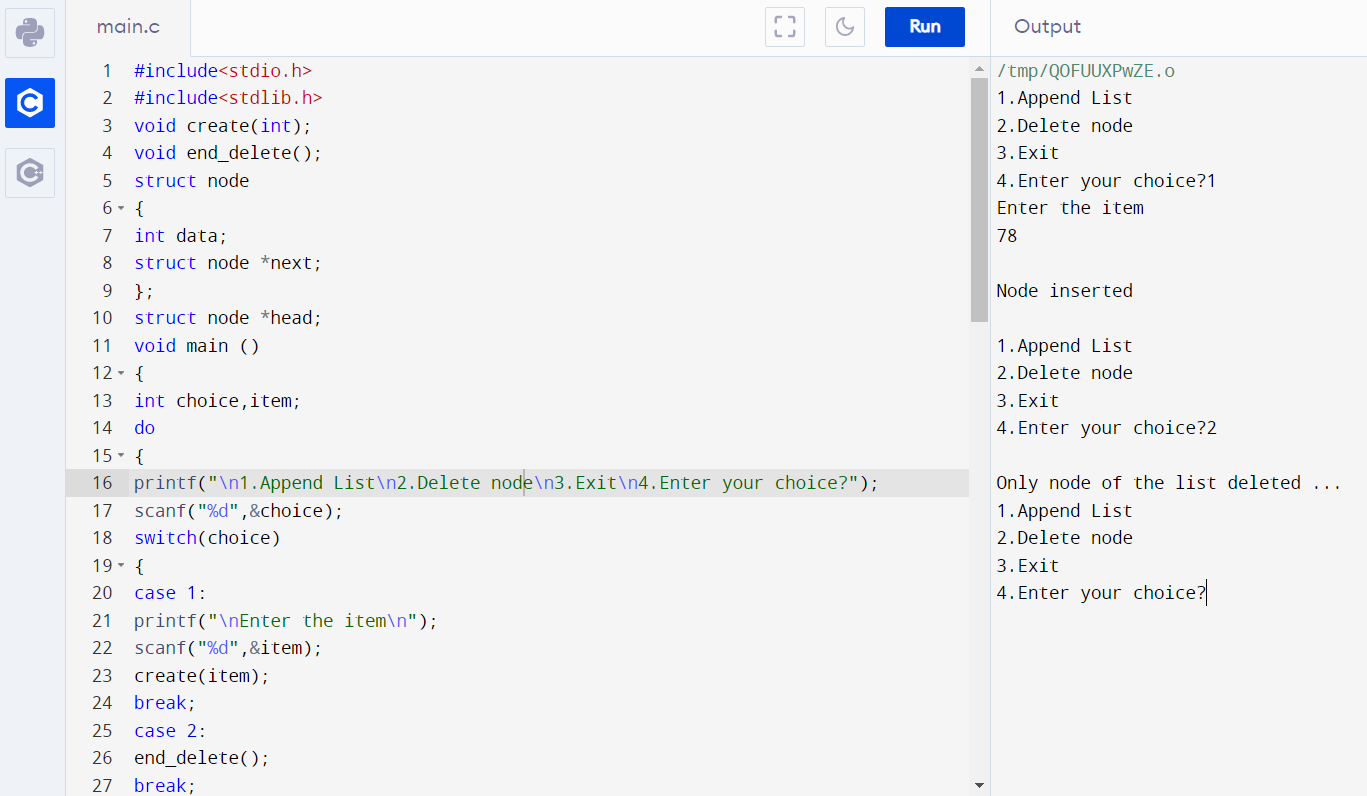
free(ptr);

printf("\n Deleted Node from the last ...");

}

}

**Output:**



**Q1G. Write a program to delete a node at any location**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void create(int);

void delete\_specified();

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice,item;

do

{

printf("\n1.Append List\n2.Delete node\n3.Exit\n4.Enter your choice?");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the item\n");

scanf("%d",&item);

create(item);

break;

case 2:

delete\_specified();

break;

case 3:

exit(0);

break;

default:

printf("\nPlease enter valid choice\n");

}

}while(choice != 3);

}

void create(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

}

else

{

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nNode inserted\n");

}

}

void delete\_specified()

{

struct node \*ptr, \*ptr1;

int loc,i;

scanf("%d",&loc);

ptr=head;

for(i=0;i<loc;i++)

{

ptr1 = ptr;

ptr = ptr->next;

if(ptr == NULL)

{

printf("\nThere are less than %d elements in the list..\n",loc);

return;

}

}

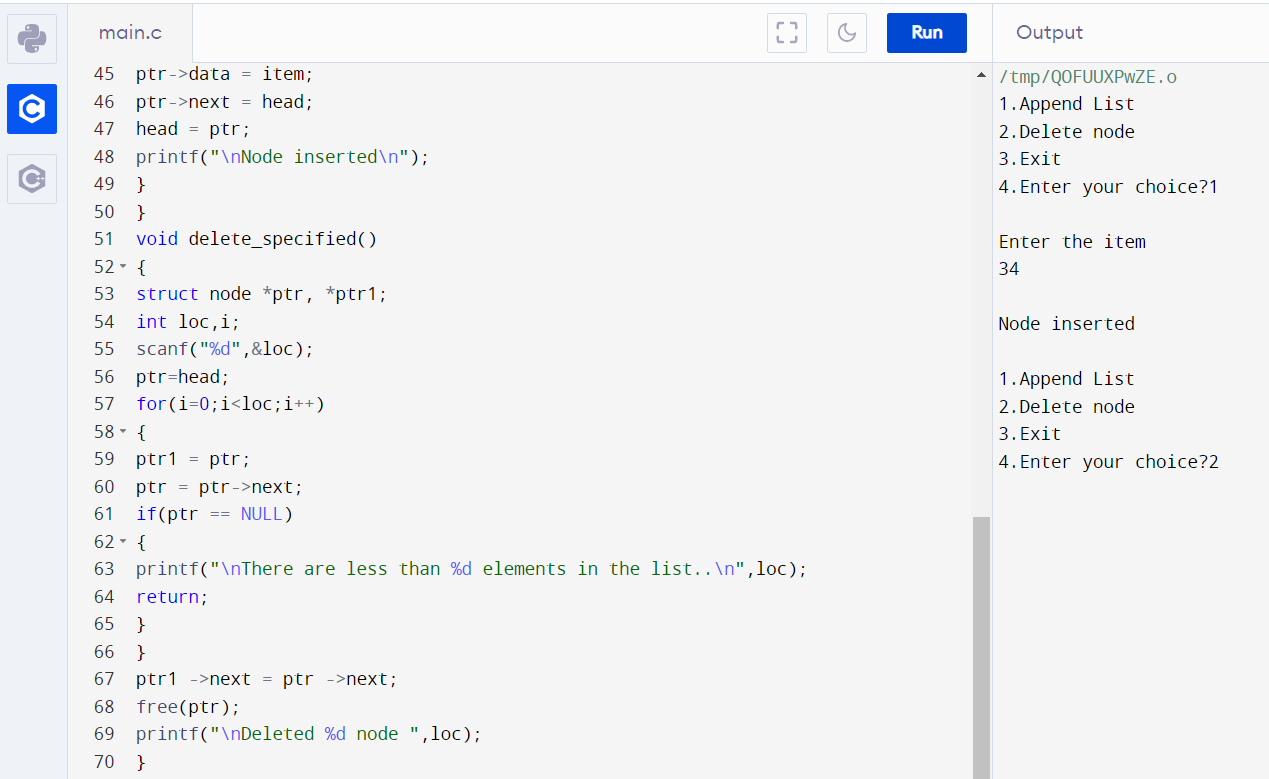
ptr1 ->next = ptr ->next;

free(ptr);

printf("\nDeleted %d node ",loc);

}

**Output:**



**Q1H. Write a program to search an existing element**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void create(int);

void search();

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void main ()

{

int choice,item,loc;

do

{

printf("\n1.Create\n2.Search\n3.Exit\n4.Enter your choice?");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the item\n");

scanf("%d",&item);

create(item);

break;

case 2:

search();

case 3:

exit(0);

break;

default:

printf("\nPlease enter valid choice\n");

}

}while(choice != 3);

}

void create(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

}

else

{

ptr->data = item;

ptr->next = head;

head = ptr;

printf("\nNode inserted\n");

}

}

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item which you want to search?\n");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("item found at location %d ",i+1);

flag=0;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

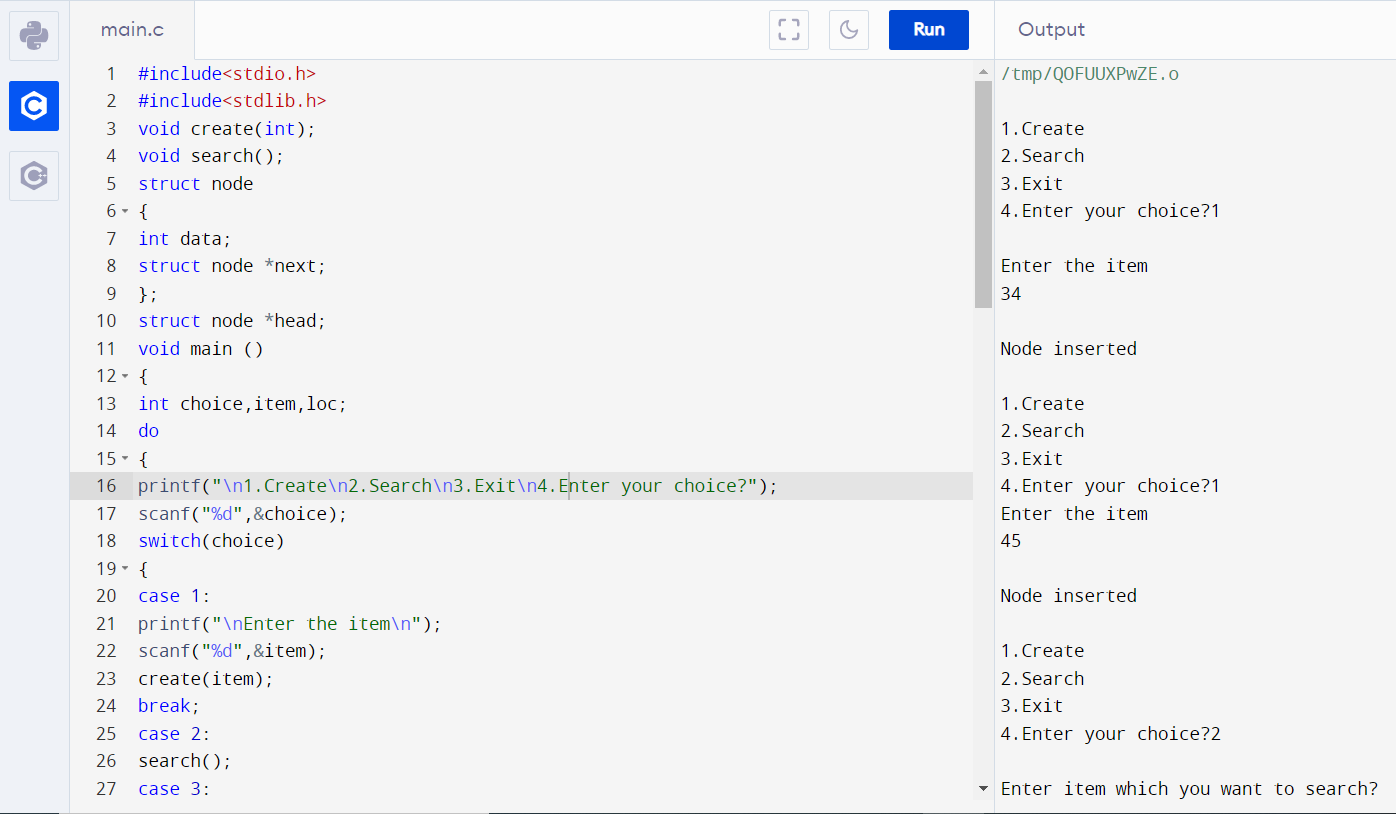
printf("Item not found\n");

}

}

}

**Output:**



**Q1I. Write a program to display all elements**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \*nextptr;

}\*stnode;

void createNodeList(int n);

void displayList();

int main()

{

int n;

printf("\n\n Linked List : To create and display Singly Linked List :\n");

printf(" Input the number of nodes : ");

scanf("%d", &n);

createNodeList(n);

printf("\n Data entered in the list : \n");

displayList();

return 0;

}

void createNodeList(int n)

{

struct node \*fnNode, \*tmp;

int num, i;

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

if(stnode == NULL)

{

printf(" Memory can not be allocated.");

}

else

{

printf(" Input data for node 1 : ");

scanf("%d", &num);

stnode->num = num;

stnode->nextptr = NULL; // links the address field to NULL

tmp = stnode;

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

{

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

if(fnNode == NULL)

{

printf(" Memory can not be allocated.");

break;

}

else

{

printf(" Input data for node %d : ", i);

scanf(" %d", &num);

fnNode->num = num;

fnNode->nextptr = NULL;

tmp->nextptr = fnNode;

tmp = tmp->nextptr;

}

}

}

}

void displayList()

{

struct node \*tmp;

if(stnode == NULL)

{

printf(" List is empty.");

}

else

{

tmp = stnode;

while(tmp != NULL)

{

printf(" Data = %d\n", tmp->num);

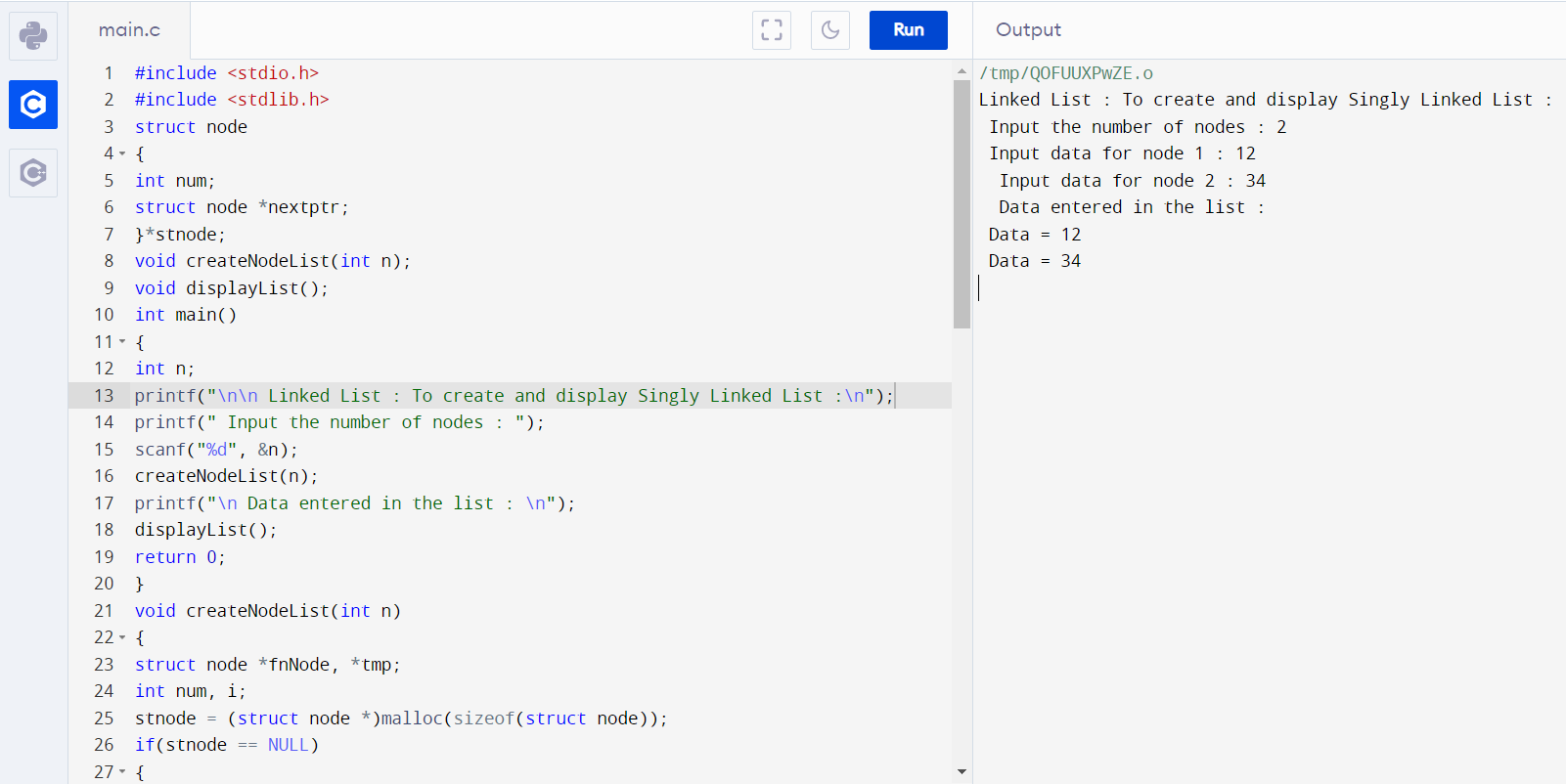
tmp = tmp->nextptr;

}

}

}

**Output:**



**DOUBLY LINKED LIST OPERATIONS:**

**Q2A. Write a program to create a doubly linked list of n nodes**

**Ans – C Program:**

#include <stdio.h>

struct node

{

int data;

struct node \*previous;

struct node \*next;

};

struct node \*head, \*tail = NULL;

void addNode(int data)

{

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

newNode->data = data;

if(head == NULL)

{

head = tail = newNode;

head->previous = NULL;

tail->next = NULL;

}

else

{

tail->next = newNode;

newNode->previous = tail;

tail = newNode;

tail->next = NULL;

}

}

void display()

{

struct node \*current = head;

if(head == NULL)

{

printf("List is empty\n");

return;

}

printf("Nodes of doubly linked list: \n");

while(current != NULL)

{

printf("%d ", current->data);

current = current->next;

}

}

int main()

{

addNode(1);

addNode(2);

addNode(3);

addNode(4);

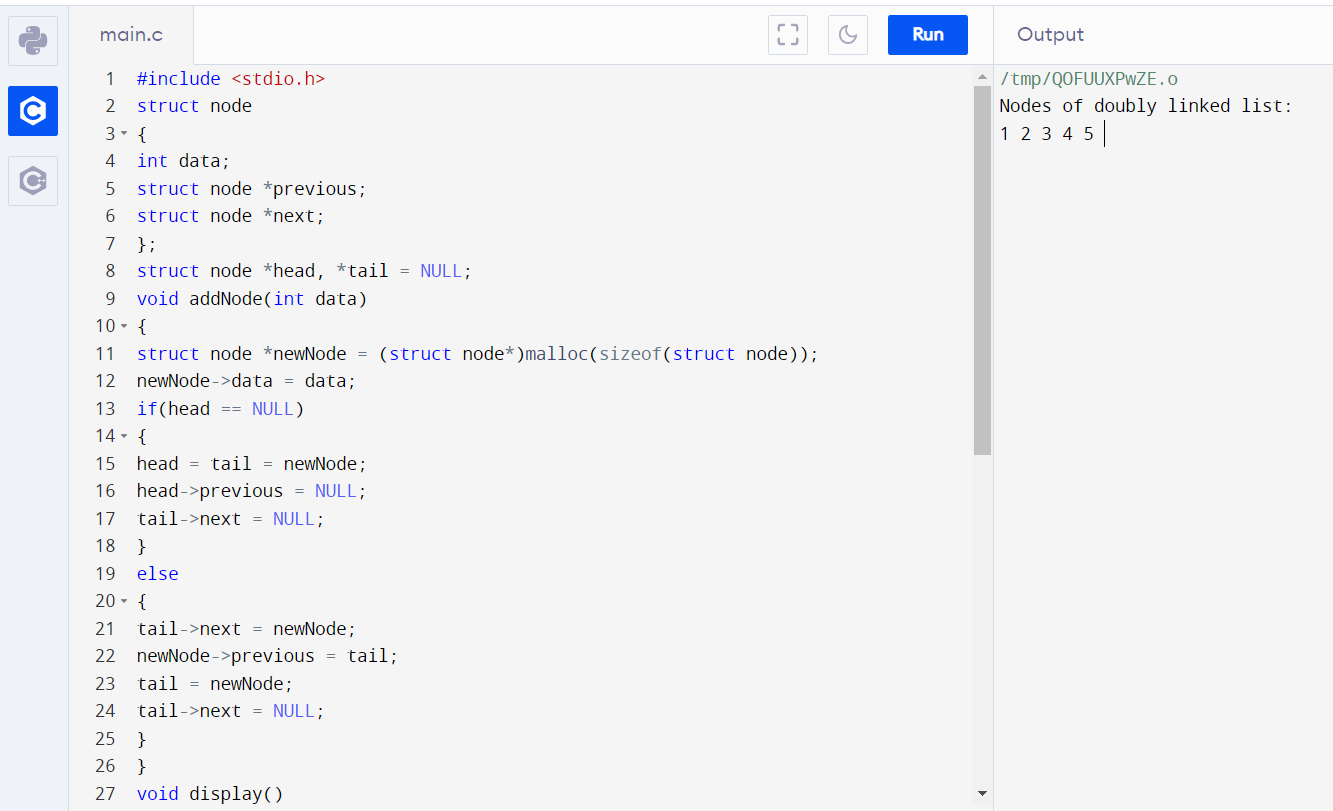
addNode(5);

display();

return 0;

}

**Output:**



**Q2B. Write a program to insert a new node at the beginning**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

void DlLinsertNodeAtBeginning(int num);

void displayDlList(int a);

int main()

{

int n,num1,a;

stnode = NULL;

ennode = NULL;

printf(" Input the number of nodes : ");

scanf("%d", &n);

DlListcreation(n);

a=1;

displayDlList(a);

printf(" Input data for the first node : ");

scanf("%d", &num1);

DlLinsertNodeAtBeginning(num1);

a=2;

displayDlList(a);

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : "); // assigning data in the first node

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

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

{

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

if(fnNode != NULL)

{

printf(" Input data for node %d : ", i);

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode;

fnNode->nextptr = NULL;

ennode->nextptr = fnNode;

ennode = fnNode;

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

void DlLinsertNodeAtBeginning(int num)

{

struct node \* newnode;

if(stnode == NULL)

{

printf(" No data found in the list!\n");

}

else

{

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

newnode->num = num;

newnode->nextptr = stnode;

newnode->preptr = NULL;

stnode->preptr = newnode;

stnode = newnode;

}

}

void displayDlList(int m)

{

struct node \* tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

if (m==1)

{

printf("\n Data entered in the list are :\n");

}

else

{

printf("\n After insertion the new list are :\n");

}

while(tmp != NULL)

{

printf(" node %d : %d\n", n, tmp->num);

n++;

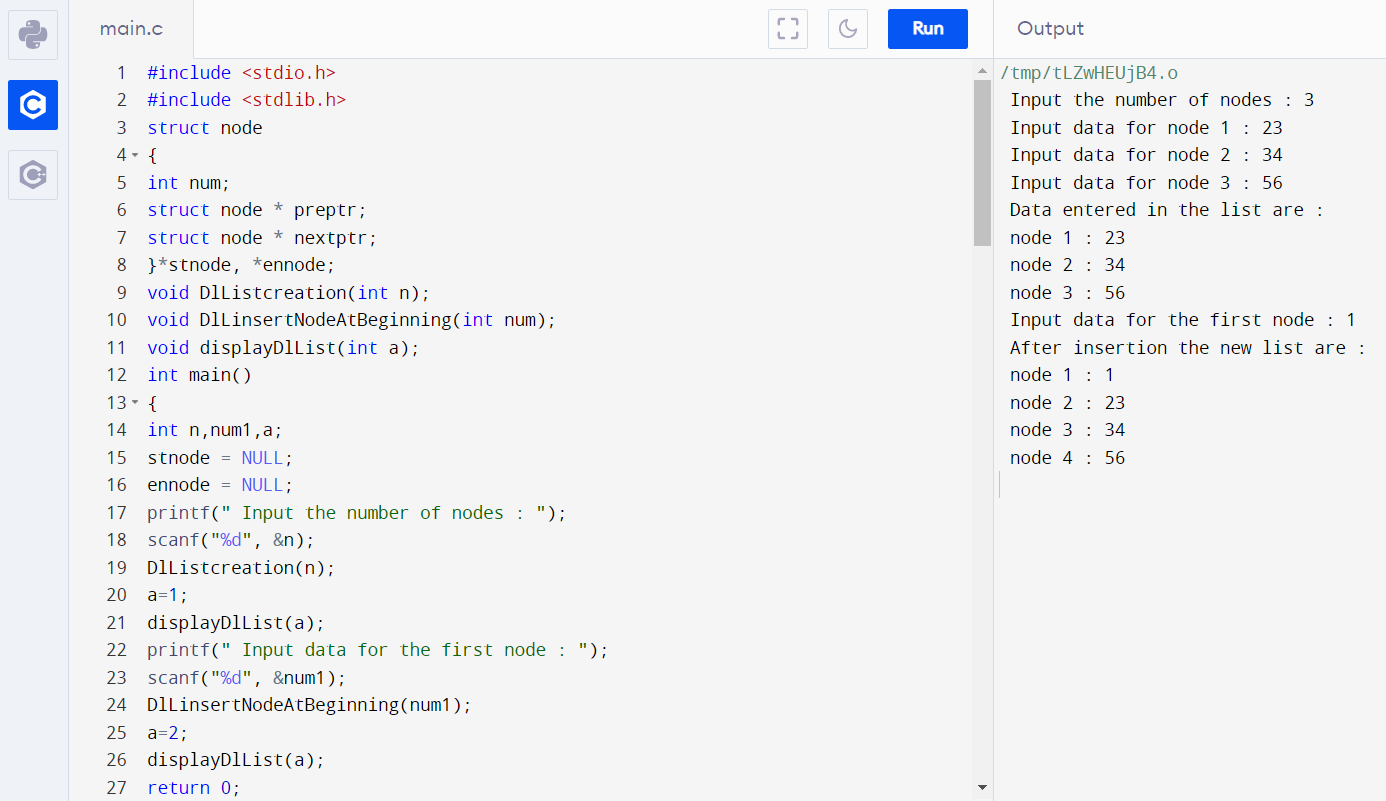
tmp = tmp->nextptr;

}

}

}

**Output:**



**Q2C. Write a program to insert a new node at the end**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

void DlLinsertNodeAtEnd(int num);

void displayDlList(int a);

int main()

{

int n,num1,a;

stnode = NULL;

ennode = NULL;

printf(" Input the number of nodes : ");

scanf("%d", &n);

DlListcreation(n);

a=1;

displayDlList(a);

printf(" Input data for the last node : ");

scanf("%d", &num1);

DlLinsertNodeAtEnd(num1);

a=2;

displayDlList(a);

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : "); // assigning data in the first node

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

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

{

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

if(fnNode != NULL)

{

printf(" Input data for node %d : ", i);

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode;

fnNode->nextptr = NULL;

ennode->nextptr = fnNode;

ennode = fnNode;

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

void DlLinsertNodeAtEnd(int num)

{

struct node \* newnode;

if(ennode == NULL)

{

printf(" No data found in the list!\n");

}

else

{

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

newnode->num = num;

newnode->nextptr = NULL;

newnode->preptr = ennode;

ennode->nextptr = newnode;

ennode = newnode;

}

}

void displayDlList(int m)

{

struct node \* tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

if (m==1)

{

printf("\n Data entered in the list are :\n");

}

else

{

printf("\n After insertion the new list are :\n");

}

while(tmp != NULL)

{

printf(" node %d : %d\n", n, tmp->num);

n++;

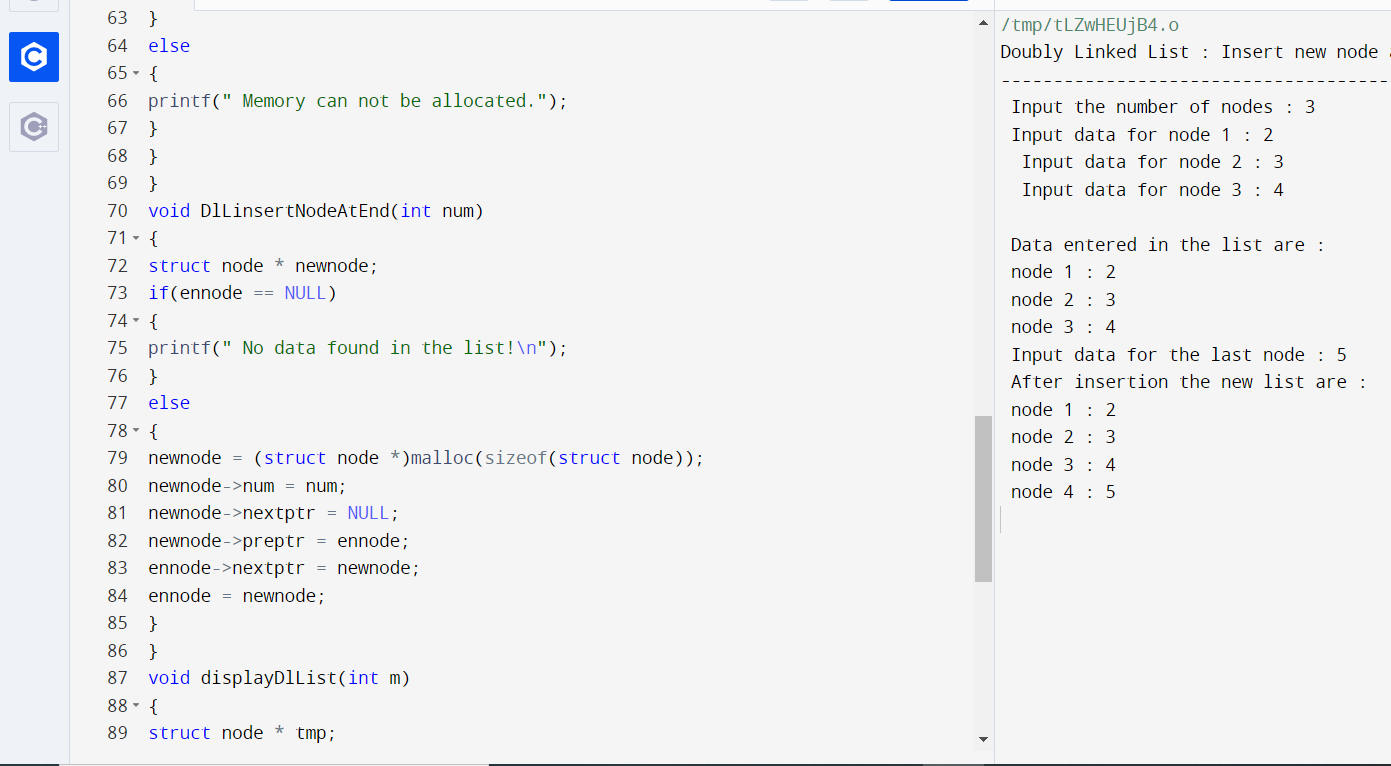
tmp = tmp->nextptr;

}

}

}

**Output:**



**Q2D**. **Write a program to insert a new node at any location**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

void DlLinsertNodeAtBeginning(int num);

void DlLinsertNodeAtEnd(int num);

void DlLinsertNodeAtAny(int num, int pos);

void displayDlList(int a);

int main()

{

int n,num1,a,insPlc;

stnode = NULL;

ennode = NULL;

printf(" Input the number of nodes : ");

scanf("%d", &n);

DlListcreation(n);

a=1;

displayDlList(a);

printf(" Input the position ( 1 to %d ) to insert a new node : ",n+1);

scanf("%d", &insPlc);

printf(" Input data for the position %d : ", insPlc);

scanf("%d", &num1);

DlLinsertNodeAtAny(num1,insPlc);

a=2;

displayDlList(a);

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : ");

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

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

{

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

if(fnNode != NULL)

{

printf(" Input data for node %d : ", i);

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode;

fnNode->nextptr = NULL;

ennode->nextptr = fnNode;

ennode = fnNode;

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

void DlLinsertNodeAtAny(int num, int pos)

{

int i;

struct node \* newnode, \*tmp;

if(ennode == NULL)

{

printf(" No data found in the list!\n");

}

else

{

tmp = stnode;

i=1;

while(i<pos-1 && tmp!=NULL)

{

tmp = tmp->nextptr;

i++;

}

if(pos == 1)

{

DlLinsertNodeAtBeginning(num);

}

else if(tmp == ennode)

{

DlLinsertNodeAtEnd(num);

}

else if(tmp!=NULL)

{

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

newnode->num = num;

newnode->nextptr = tmp->nextptr;

newnode->preptr = tmp;

if(tmp->nextptr != NULL)

{

tmp->nextptr->preptr = newnode;

}

tmp->nextptr = newnode;

}

else

{

printf(" The position you entered, is invalid.\n");

}

}

}

void DlLinsertNodeAtBeginning(int num)

{

struct node \* newnode;

if(stnode == NULL)

{

printf(" No data found in the list!\n");

}

else

{

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

newnode->num = num;

newnode->nextptr = stnode;

newnode->preptr = NULL;

stnode->preptr = newnode;

stnode = newnode;

}

}

void DlLinsertNodeAtEnd(int num)

{

struct node \* newnode;

if(ennode == NULL)

{

printf(" No data found in the list!\n");

}

else

{

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

newnode->num = num;

newnode->nextptr = NULL;

newnode->preptr = ennode;

ennode->nextptr = newnode;

ennode = newnode;

}

}

void displayDlList(int m)

{

struct node \* tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

if (m==1)

{

printf("\n Data entered in the list are :\n");

}

else

{

printf("\n After insertion the new list are :\n");

}

while(tmp != NULL)

{

printf(" node %d : %d\n", n, tmp->num);

n++;

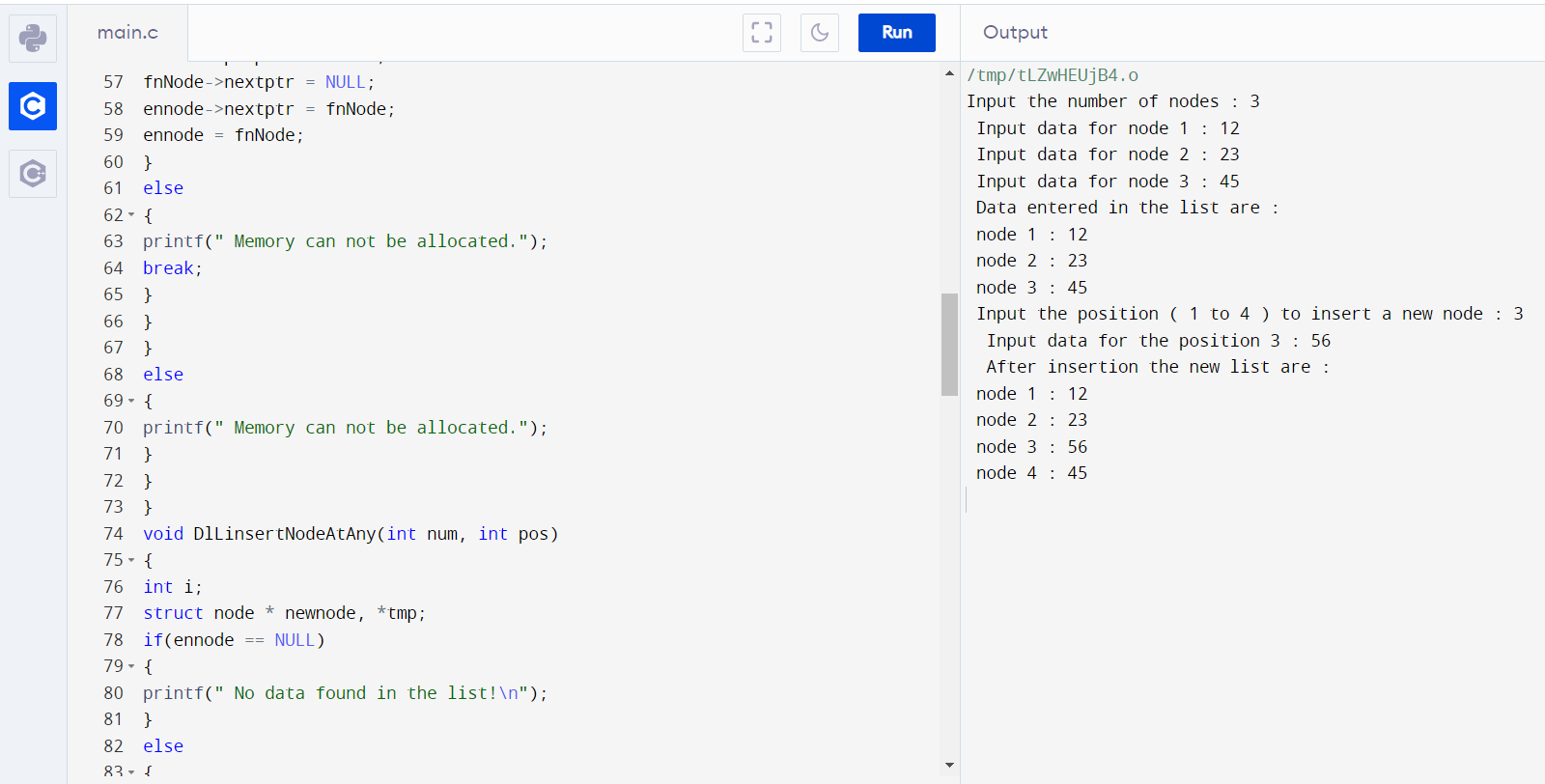
tmp = tmp->nextptr;

}

}

}

**Output:**



**Q2E. Write a program to delete a node at the beginning**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

void DlListDeleteFirstNode();

void displayDlList(int a);

int main()

{

int n,num1,a,insPlc;

stnode = NULL;

ennode = NULL;

printf(" Input the number of nodes (3 or more ): ");

scanf("%d", &n);

DlListcreation(n);

a=1;

displayDlList(a);

DlListDeleteFirstNode();

a=2;

displayDlList(a);

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : ");

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

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

{

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

if(fnNode != NULL)

{

printf(" Input data for node %d : ", i);

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode;

fnNode->nextptr = NULL;

ennode->nextptr = fnNode;

ennode = fnNode;

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

void DlListDeleteFirstNode()

{

struct node \* NodeToDel;

if(stnode == NULL)

{

printf(" Delete is not possible. No data in the list.\n");

}

else

{

NodeToDel = stnode;

stnode = stnode->nextptr;

stnode->preptr = NULL;

free(NodeToDel);

}

}

void displayDlList(int m)

{

struct node \* tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

if (m==1)

{

printf("\n Data entered in the list are :\n");

}

else

{

printf("\n After deletion the new list are :\n");

}

while(tmp != NULL)

{

printf(" node %d : %d\n", n, tmp->num);

n++;

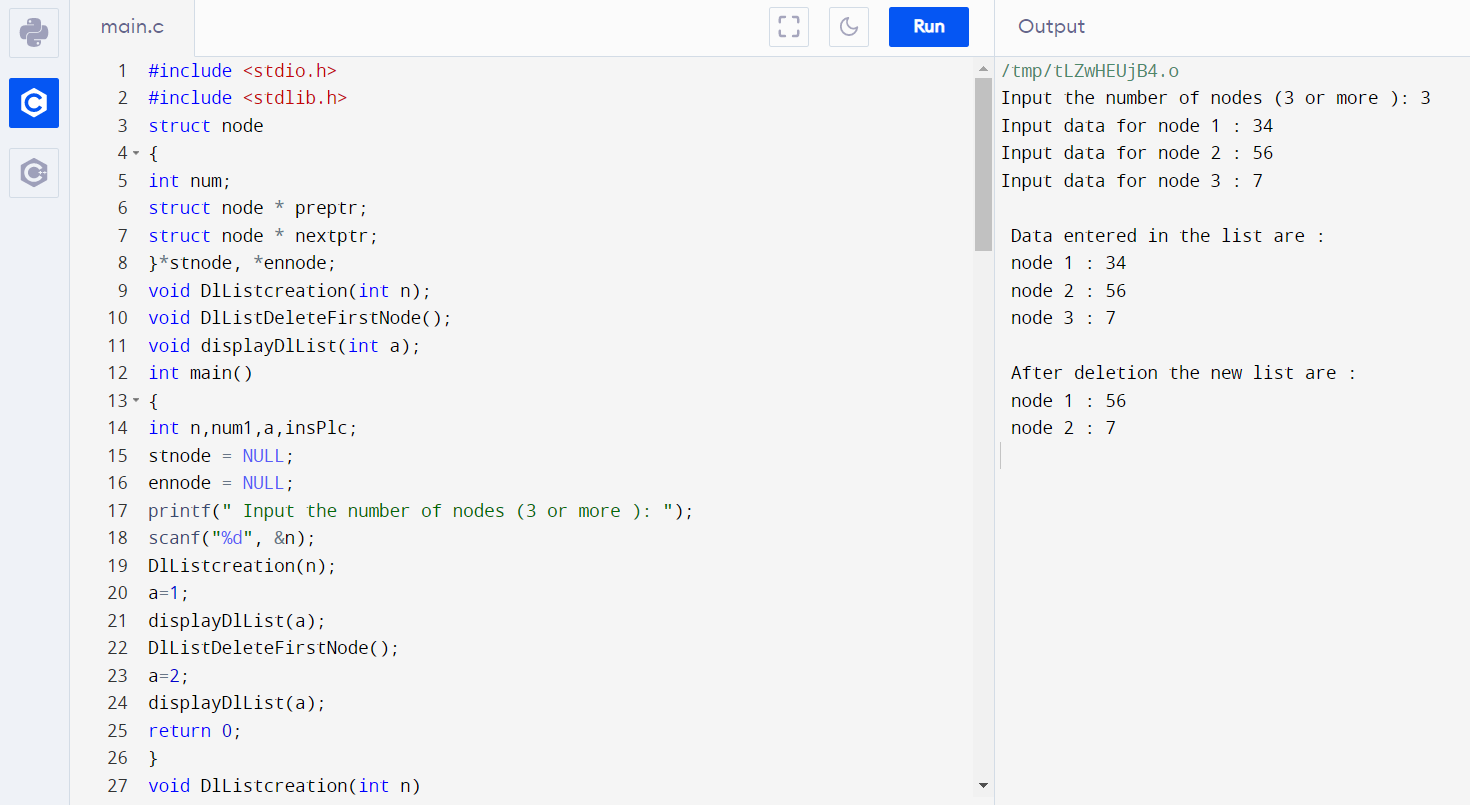
tmp = tmp->nextptr;

}

}

}

**Output:**



**Q2F. Write a program to delete a node at the end**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

void DlListDeleteLastNode();

void displayDlList(int a);

int main()

{

int n,num1,a,insPlc;

stnode = NULL;

ennode = NULL;

printf(" Input the number of nodes (3 or more ): ");

scanf("%d", &n);

DlListcreation(n);

a=1;

displayDlList(a);

DlListDeleteLastNode();

a=2;

displayDlList(a);

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

printf(" Input data for node 1 : ");

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

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

{

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

if(fnNode != NULL)

{

printf(" Input data for node %d : ", i);

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode;

fnNode->nextptr = NULL;

ennode->nextptr = fnNode;

ennode = fnNode;

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

void DlListDeleteLastNode()

{

struct node \* NodeToDel;

if(ennode == NULL)

{

printf(" Delete is not possible. No data in the list.\n");

}

else

{

NodeToDel = ennode;

ennode = ennode->preptr;

ennode->nextptr = NULL;

free(NodeToDel);

}

}

void displayDlList(int m)

{

struct node \* tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

if (m==1)

{

printf("\n Data entered in the list are :\n");

}

else

{

printf("\n After deletion the new list are :\n");

}

while(tmp != NULL)

{

printf(" node %d : %d\n", n, tmp->num);

n++;

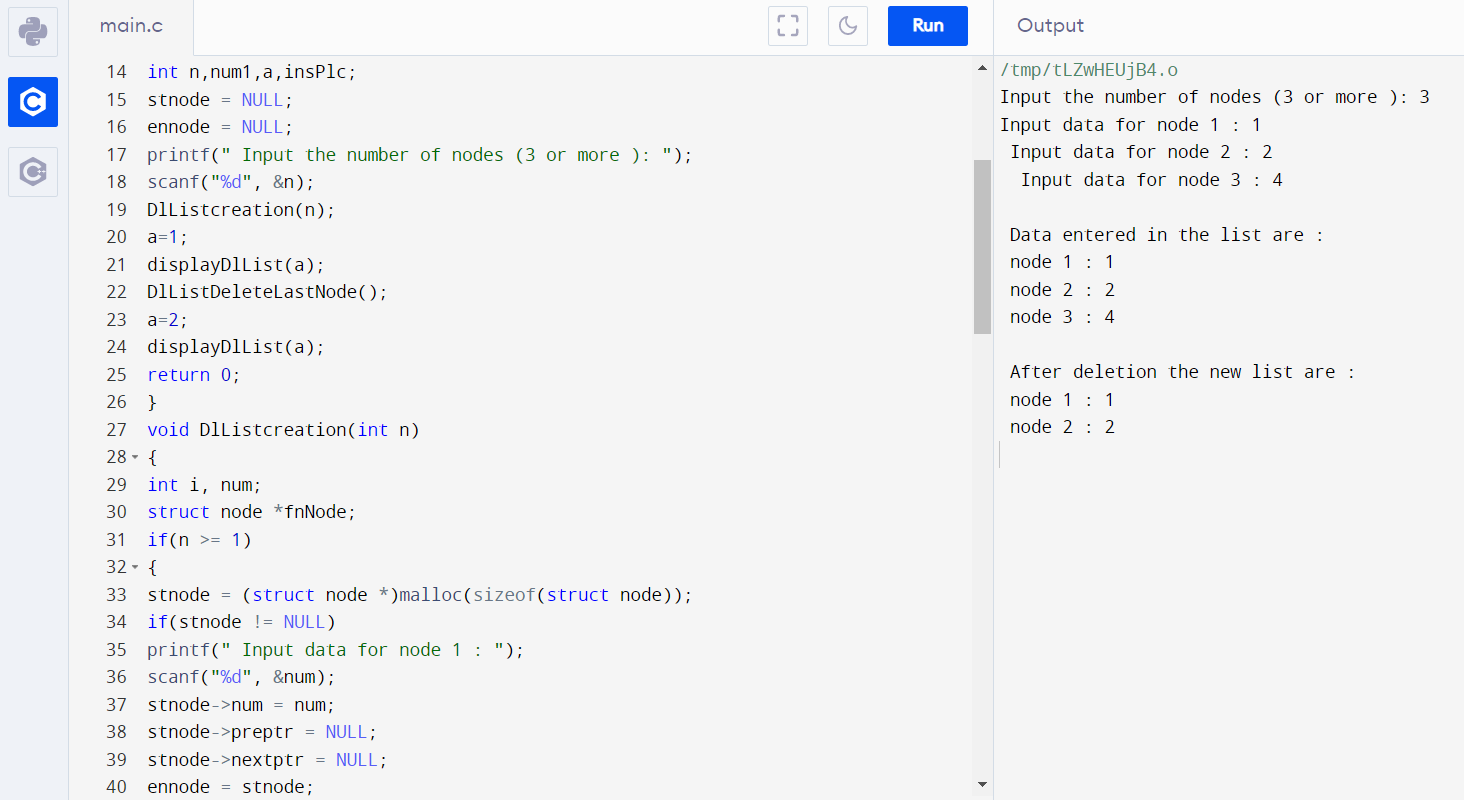
tmp = tmp->nextptr;

}

}

}

**Output:**



**Q2G. Write a program to delete a node at any location**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

void DlListDeleteFirstNode();

void DlListDeleteLastNode();

void DlListDeleteAnyNode(int pos);

void displayDlList(int a);

int main()

{

int n,num1,a,insPlc;

stnode = NULL;

ennode = NULL;

printf(" Input the number of nodes (3 or more ): ");

scanf("%d", &n);

DlListcreation(n);

a=1;

displayDlList(a);

printf(" Input the position ( 1 to %d ) to delete a node : ",n);

scanf("%d", &insPlc);

if(insPlc<1 || insPlc>n)

{

printf("\n Invalid position. Try again.\n ");

}

if(insPlc>=1 && insPlc<=n)

{

DlListDeleteAnyNode(insPlc);

a=2;

displayDlList(a);

}

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : "); // assigning data in the first node

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

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

{

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

if(fnNode != NULL)

{

printf(" Input data for node %d : ", i);

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode;

fnNode->nextptr = NULL;

ennode->nextptr = fnNode;

ennode = fnNode;

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

void DlListDeleteAnyNode(int pos)

{

struct node \*curNode;

int i;

curNode = stnode;

for(i=1; i<pos && curNode!=NULL; i++)

{

curNode = curNode->nextptr;

}

if(pos == 1)

{

DlListDeleteFirstNode();

}

else if(curNode == ennode)

{

DlListDeleteLastNode();

}

else if(curNode != NULL)

{

curNode->preptr->nextptr = curNode->nextptr;

curNode->nextptr->preptr = curNode->preptr;

free(curNode);

}

else

{

printf(" The given position is invalid!\n");

}

}

void DlListDeleteFirstNode()

{

struct node \* NodeToDel;

if(stnode == NULL)

{

printf(" Delete is not possible. No data in the list.\n");

}

else

{

NodeToDel = stnode;

stnode = stnode->nextptr;

stnode->preptr = NULL;

free(NodeToDel);

}

}

void DlListDeleteLastNode()

{

struct node \* NodeToDel;

if(ennode == NULL)

{

printf(" Delete is not possible. No data in the list.\n");

}

else

{

NodeToDel = ennode;

ennode = ennode->preptr;

ennode->nextptr = NULL;

free(NodeToDel);

}

}

void displayDlList(int m)

{

struct node \* tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

if (m==1)

{

printf("\n Data entered in the list are :\n");

}

else

{

printf("\n After deletion the new list are :\n");

}

while(tmp != NULL)

{

printf(" node %d : %d\n", n, tmp->num);

n++;

tmp = tmp->nextptr;

}

}

}

**Output:**



**Q2H. Write a program to search an existing element**

**Ans – C Program:**

#include<stdio.h>

#include<stdlib.h>

void create(int);

void search();

struct node

{

int data;

struct node \*next;

struct node \*prev;

};

struct node \*head;

void main ()

{

int choice,item,loc;

do

{

printf("\n1.Create\n2.Search\n3.Exit\n4.Enter your choice?");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the item\n");

scanf("%d",&item);

create(item);

break;

case 2:

search();

case 3:

exit(0);

break;

default:

printf("\nPlease enter valid choice\n");

}

}

while(choice != 3);

}

void create(int item)

{

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

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

if(head==NULL)

{

ptr->next = NULL;

ptr->prev=NULL;

ptr->data=item;

head=ptr;

}

else

{

ptr->data=item;printf("\nPress 0 to insert more ?\n");

ptr->prev=NULL;

ptr->next = head;

head->prev=ptr;

head=ptr;

}

printf("\nNode Inserted\n");

}

}

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("\nEmpty List\n");

}

else

{

printf("\nEnter item which you want to search?\n");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("\nitem found at location %d ",i+1);

flag=0;

break;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

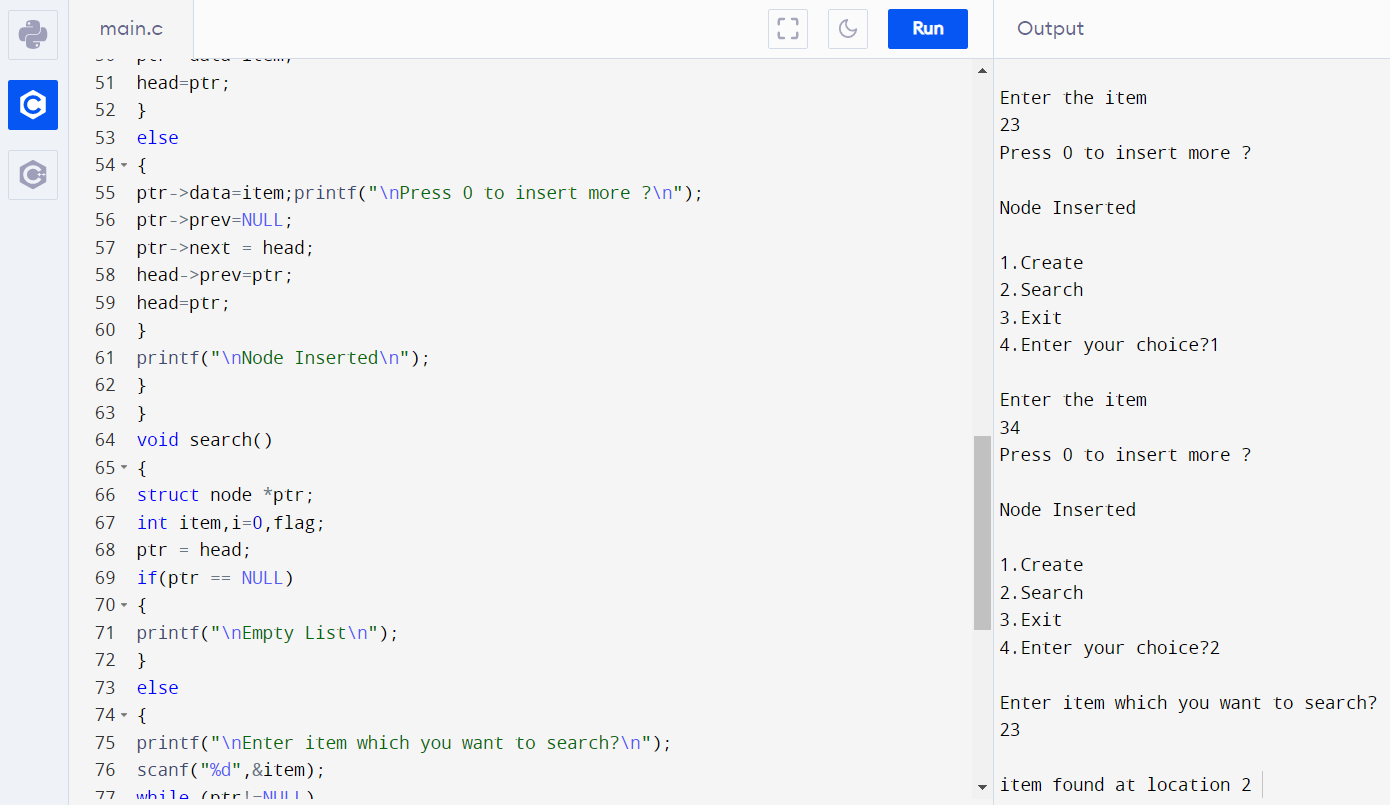
printf("\nItem not found\n");

}

}

}

**Output:**



**Q2I. Write a program to display all elements**

**Ans – C Program:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num;

struct node \* preptr;

struct node \* nextptr;

}\*stnode, \*ennode;

void DlListcreation(int n);

void displayDlList();

int main()

{

int n;

stnode = NULL;

ennode = NULL;

printf(" Input the number of nodes : ");

scanf("%d", &n);

DlListcreation(n);

displayDlList();

return 0;

}

void DlListcreation(int n)

{

int i, num;

struct node \*fnNode;

if(n >= 1)

{

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

if(stnode != NULL)

{

printf(" Input data for node 1 : ");

scanf("%d", &num);

stnode->num = num;

stnode->preptr = NULL;

stnode->nextptr = NULL;

ennode = stnode;

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

{

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

if(fnNode != NULL)

{

printf(" Input data for node %d : ", i);

scanf("%d", &num);

fnNode->num = num;

fnNode->preptr = ennode;

fnNode->nextptr = NULL;

ennode->nextptr = fnNode;

ennode = fnNode;

}

else

{

printf(" Memory can not be allocated.");

break;

}

}

}

else

{

printf(" Memory can not be allocated.");

}

}

}

void displayDlList()

{

struct node \* tmp;

int n = 1;

if(stnode == NULL)

{

printf(" No data found in the List yet.");

}

else

{

tmp = stnode;

printf("\n\n Data entered on the list are :\n");

while(tmp != NULL)

{

printf(" node %d : %d\n", n, tmp->num);

n++;

tmp = tmp->nextptr;

}

}

}

**Output:**

