**COMPETITIVE PROGRAMMING**

**LINKEDLIST SET2**

**AM.SC.P2CSC19034**

**SRUTHY P R**

**DOUBLYLINKEDLIST:**

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

struct node

{

int info;

struct node \*next;

struct node \*prev;

}\*head;

class double\_llist

{

public:

void create\_list(int value);

void add\_begin(int value);

void add\_after(int value, int position);

void delete\_element(int value);

void search\_element(int value);

void display\_dlist();

double\_llist()

{

head = NULL;

}

};

void double\_llist::create\_list(int value)

{

struct node \*s, \*temp;

temp = new(struct node);

temp->info = value;

temp->next = NULL;

if (head == NULL)

{

temp->prev = NULL;

head = temp;

}

else

{

s = head;

while (s->next != NULL)

s = s->next;

s->next = temp;

temp->prev = s;

}

}

void double\_llist::add\_begin(int value)

{

if (head == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*temp;

temp = new(struct node);

temp->prev = NULL;

temp->info = value;

temp->next = head;

head->prev = temp;

head = temp;

cout<<"Element Inserted"<<endl;

}

void double\_llist::add\_after(int value, int pos)

{

if (head == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*tmp, \*q;

int i;

q = head;

for (i = 0;i < pos - 1;i++)

{

q = q->next;

if (q == NULL)

{

cout<<"There are less than ";

cout<<pos<<" elements."<<endl;

return;

}

}

tmp = new(struct node);

tmp->info = value;

if (q->next == NULL)

{

q->next = tmp;

tmp->next = NULL;

tmp->prev = q;

}

else

{

tmp->next = q->next;

tmp->next->prev = tmp;

q->next = tmp;

tmp->prev = q;

}

cout<<"Element Inserted"<<endl;

}

void double\_llist::delete\_element(int value)

{

struct node \*tmp, \*q;

if (head->info == value)

{

tmp = head;

head = head->next;

head->prev = NULL;

cout<<"Element Deleted"<<endl;

free(tmp);

return;

}

q = head;

while (q->next->next != NULL)

{

if (q->next->info == value)

{

tmp = q->next;

q->next = tmp->next;

tmp->next->prev = q;

cout<<"Element Deleted"<<endl;

free(tmp);

return;

}

q = q->next;

}

if (q->next->info == value)

{

tmp = q->next;

free(tmp);

q->next = NULL;

cout<<"Element Deleted"<<endl;

return;

}

cout<<"Element "<<value<<" not found"<<endl;

}

void double\_llist::display\_dlist()

{

struct node \*q;

if (head == NULL)

{

cout<<"List empty,nothing to display"<<endl;

return;

}

q = head;

cout<<"The Doubly Link List is :"<<endl;

while (q != NULL)

{

cout<<q->info<<" <-> ";

q = q->next;

}

cout<<"NULL"<<endl;

}

int main()

{

int choice, element, position;

double\_llist dl;

while (1)

{

cout<<endl<<"Operations on Doubly linked list"<<endl;

cout<<"1.Create Node"<<endl;

cout<<"2.Add at begining"<<endl;

cout<<"3.Add after position"<<endl;

cout<<"4.Delete"<<endl;

cout<<"5.Display"<<endl;

cout<<"6.Quit"<<endl;

cout<<"Enter your choice : ";

cin>>choice;

switch ( choice )

{

case 1:

cout<<"Enter the element: ";

cin>>element;

dl.create\_list(element);

cout<<endl;

break;

case 2:

cout<<"Enter the element: ";

cin>>element;

dl.add\_begin(element);

cout<<endl;

break;

case 3:

cout<<"Enter the element: ";

cin>>element;

cout<<"Insert Element after postion: ";

cin>>position;

dl.add\_after(element, position);

cout<<endl;

break;

case 4:

if (head == NULL)

{

cout<<"List empty,nothing to delete"<<endl;

break;

}

cout<<"Enter the element for deletion: ";

cin>>element;

dl.delete\_element(element);

cout<<endl;

break;

case 5:

dl.display\_dlist();

cout<<endl;

break;

case 6:

exit(1);

default:

cout<<"Wrong choice"<<endl;

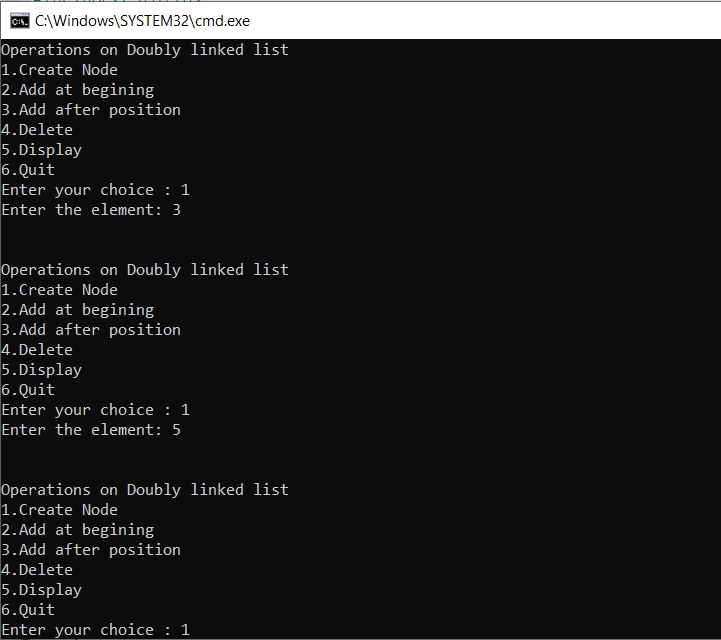
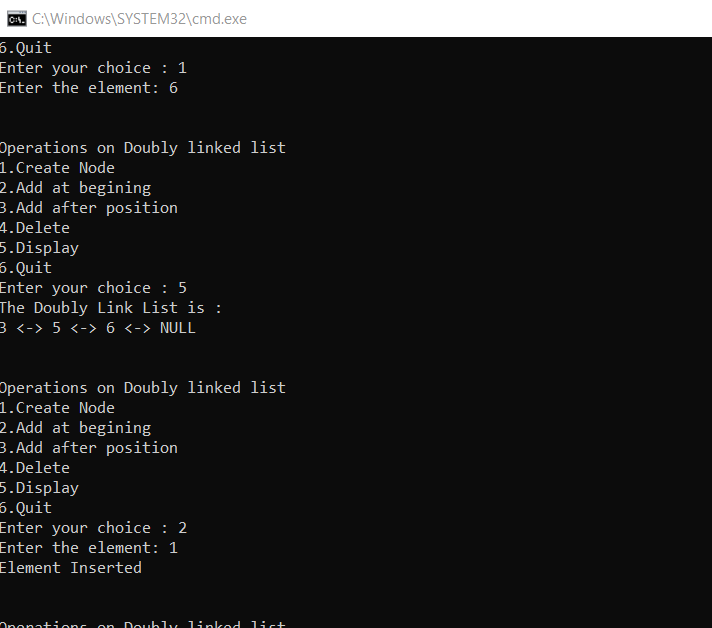
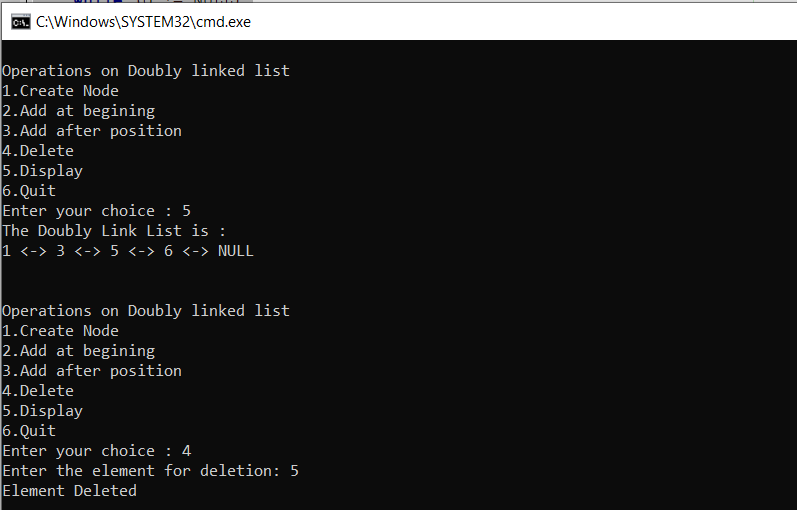
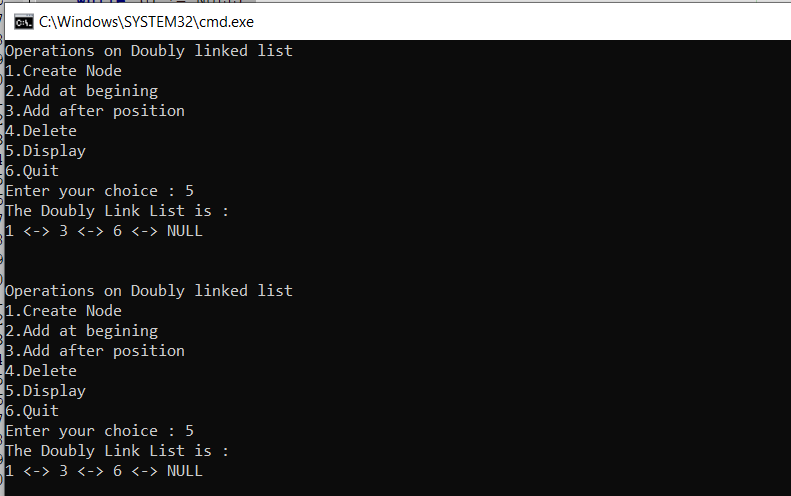
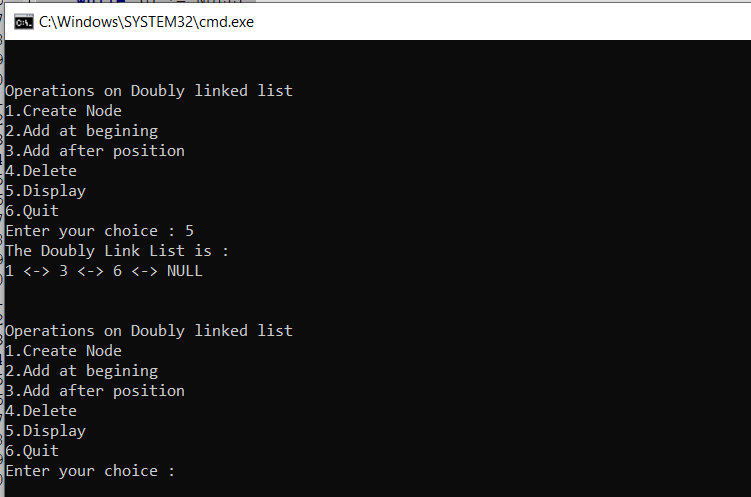
}

}

return 0;

}

**OUTPUT:**



**CIRCULARLINKEDLIST:**

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

struct node

{

int info;

struct node \*next;

}\*last;

class circular\_llist

{

public:

void create\_node(int value);

void add\_begin(int value);

void add\_after(int value, int position);

void delete\_element(int value);

void display\_list();

circular\_llist()

{

last = NULL;

}

};

void circular\_llist::create\_node(int value)

{

struct node \*temp;

temp = new(struct node);

temp->info = value;

if (last == NULL)

{

last = temp;

temp->next = last;

}

else

{

temp->next = last->next;

last->next = temp;

last = temp;

}

}

void circular\_llist::add\_begin(int value)

{

if (last == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*temp;

temp = new(struct node);

temp->info = value;

temp->next = last->next;

last->next = temp;

}

void circular\_llist::add\_after(int value, int pos)

{

if (last == NULL)

{

cout<<"First Create the list."<<endl;

return;

}

struct node \*temp, \*s;

s = last->next;

for (int i = 0;i < pos-1;i++)

{

s = s->next;

if (s == last->next)

{

cout<<"There are less than ";

cout<<pos<<" in the list"<<endl;

return;

}

}

temp = new(struct node);

temp->next = s->next;

temp->info = value;

s->next = temp;

if (s == last)

{

last=temp;

}

}

void circular\_llist::delete\_element(int value)

{

struct node \*temp, \*s;

s = last->next;

if (last->next == last && last->info == value)

{

temp = last;

last = NULL;

free(temp);

return;

}

if (s->info == value)

{

temp = s;

last->next = s->next;

free(temp);

return;

}

while (s->next != last)

{

if (s->next->info == value)

{

temp = s->next;

s->next = temp->next;

free(temp);

cout<<"Element "<<value;

cout<<" deleted from the list"<<endl;

return;

}

s = s->next;

}

if (s->next->info == value)

{

temp = s->next;

s->next = last->next;

free(temp);

last = s;

return;

}

cout<<"Element "<<value<<" not found in the list"<<endl;

}

void circular\_llist::display\_list()

{

struct node \*s;

if (last == NULL)

{

cout<<"List is empty, nothing to display"<<endl;

return;

}

s = last->next;

cout<<"Circular Link List: "<<endl;

while (s != last)

{

cout<<s->info<<"->";

s = s->next;

}

cout<<s->info<<endl;

}

int main()

{

int choice, element, position;

circular\_llist cl;

while (1)

{

cout<<endl<<"Circular singly linked list"<<endl;

cout<<"1.Create Node"<<endl;

cout<<"2.Add at beginning"<<endl;

cout<<"3.Add after"<<endl;

cout<<"4.Delete"<<endl;

cout<<"5.Display"<<endl;

cout<<"6.Quit"<<endl;

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter the element: ";

cin>>element;

cl.create\_node(element);

cout<<endl;

break;

case 2:

cout<<"Enter the element: ";

cin>>element;

cl.add\_begin(element);

cout<<endl;

break;

case 3:

cout<<"Enter the element: ";

cin>>element;

cout<<"Insert element after position: ";

cin>>position;

cl.add\_after(element, position);

cout<<endl;

break;

case 4:

if (last == NULL)

{

cout<<"List is empty, nothing to delete"<<endl;

break;

}

cout<<"Enter the element for deletion: ";

cin>>element;

cl.delete\_element(element);

cout<<endl;

break;

case 5:

cl.display\_list();

break;

case 6:

exit(1);

break;

default:

cout<<"Wrong choice"<<endl;

}

}

return 0;

}

**OUTPUT:**

