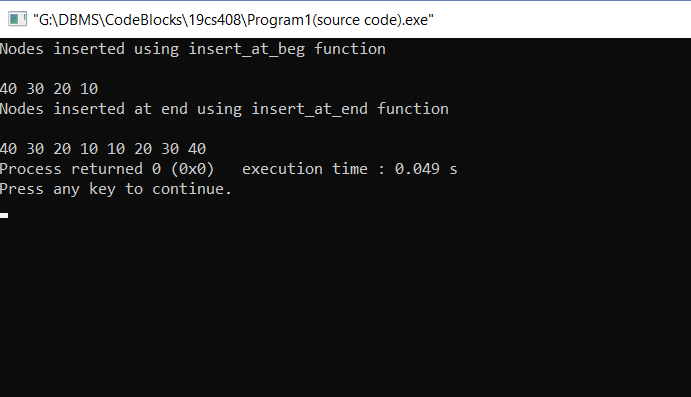
PROGRAM1:

DOUBLY LINKED LIST (XOR)

OUTPUT:



INPUT:

#include<iostream>

using namespace std;

class node

{

public:

int data;

node\* npx;

};

node\* XOR(node\* a, node\* b)

{

return (node\*) ((uintptr\_t) (a) ^ (uintptr\_t) (b));

}

void insert\_at\_begin(node \*\*first, int data)

{

node \*new\_node=new node();

new\_node->data=data;

new\_node->npx=\*first;

if(\*first!=NULL)

{

(\*first)->npx = XOR(new\_node , (\*first)->npx);

}

\*first=new\_node;

}

void insert\_at\_end(node \*\*first,int data)

{

node \*prev=NULL;

node \*curr=\*first;

node \*next=XOR(prev,curr->npx);

while(next)

{

prev=curr;

curr=next;

next=XOR(prev,curr->npx);

}

next=new node();

next->data=data;

next->npx=XOR(curr,NULL);

curr->npx=XOR(curr->npx,next);

}

void printList (node \*first)

{

node \*curr = first;

node \*prev = NULL;

node \*next;

while (curr != NULL)

{

cout<<curr->data<<" ";

next = XOR (prev, curr->npx);

prev = curr;

curr = next;

}

}

int main ()

{

node \*first = NULL;

insert\_at\_begin(&first, 10);

insert\_at\_begin(&first, 20);

insert\_at\_begin(&first, 30);

insert\_at\_begin(&first, 40);

cout<<"Nodes inserted using insert\_at\_beg function\n"<<endl;

printList (first);

insert\_at\_end(&first, 10);

insert\_at\_end(&first, 20);

insert\_at\_end(&first, 30);

insert\_at\_end(&first, 40);

cout<<"\nNodes inserted at end using insert\_at\_end function\n"<<endl;

printList(first);

return (0);

}

2.

#include<iostream>

using namespace std;

class node

{

public:

int data;

node\* next;

};

node \*reverse(node \*head,int k)

{

node\* current=head;

node\* next=NULL;

node\* pre=NULL;

int count=0;

while(current!=NULL && count<k)

{

next=current->next;

current->next=pre;

pre=current;

current=next;

count++;

}

if(next!=NULL)

head->next=reverse(next,k);

return pre;

}

void push(node\*\* head\_ref,int new\_data)

{

node\* new\_node= new node();

new\_node->data=new\_data;

new\_node->next=(\*head\_ref);

(\*head\_ref)=new\_node;

}

void printlist(node \*node)

{

while(node!=NULL)

{

cout<< node->data <<" ";

node=node->next;

}

}

int main()

{

node\* head=NULL;

push(&head,9);

push(&head,8);

push(&head,7);

push(&head,6);

push(&head,5);

push(&head,4);

push(&head,3);

push(&head,2);

push(&head,1);

cout<<"Given linked list\n"<<endl;

printlist(head);

head=reverse(head, 3);

cout<<"Reversed linked list\n"<<endl;

printlist(head);

return 0;

}

