**PROGRAM 2)** Write a program to implement the following list:  
An ordinary Doubly Linked List requires space for two address fields to store the addresses of previous and next nodes. A memory efficient version of Doubly Linked List can be created using only one space for address field with every node. This memory efficient Doubly Linked List is called XOR Linked List or Memory Efficient as the list uses bitwise XOR operation to save space for one address. In the XOR linked list, instead of storing actual memory addresses, every node stores the XOR of addresses of previous and next nodes.  
Write a separate function for insertion, deletion, and traversal of the linked list.

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <stdbool.h>

typedef struct node

{

int data;

struct node\*link;

} node;

node \*head, \*tail;

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

{

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

}

void insert(node \*\*head\_ref, int data)

{

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

newnode->data=data;

if(head==NULL){

newnode->link=NULL;

head = tail = newnode;

}

else

{

newnode->link = xor(tail,NULL);

tail->link = xor(xor(tail->link,NULL),newnode);

tail = newnode;

}

}

void traverse()

{

node \*curr=head;

node \*prev=NULL,\*next;

while(curr!=NULL)

{

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

next=xor(prev,curr->link);

prev=curr;

curr=next;

}

printf("\n");

}

int delete(bool from\_tail)

{

int data;

node \*ptr;

if(from\_tail)

{

ptr=tail;

data=ptr->data;

node \*prev=xor(ptr->link,NULL);

if(prev==NULL)

{

head=NULL;

}

else{

prev->link=xor(ptr,xor(prev->link,NULL));

}

tail=prev;

}

free(ptr);

ptr=NULL;

return data;

}

int main()

{

int data;

int n;

printf("Enter the size of the list:\n");

scanf("%d",&n);

int arr[n];

printf("INSERTION INTO LIST\n");

printf("Enter the values to be inserted into the list\n");

for(int i=0;i<n;i++)

{

scanf("%d",&arr[i]);

insert(&head, arr[i]);

printf("Successfully inserted %d\n", arr[i]);

}

traverse();

printf("\n");

int x;

printf("DELETION FROM THE LIST\n");

printf("Enter the number of values to be deleted from the end\n");

scanf("%d",&x);

for (int i=0;i<x;i++)

{

data=delete(i < x);

printf("Successfully deleted %d\n",data);

traverse();

}

return 0;

}

OUTPUT:



