**WEEK – 2**

**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.**

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

// Node structure

typedef struct Node {

int data;

struct Node\* xor\_ptr; // XOR of previous and next node addresses

} Node;

// Function to perform XOR operation on pointers

Node\* xor(Node\* a, Node\* b) {

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

}

// Function to insert a node at the beginning of the list

void insert(Node\*\* head\_ref, int data) {

// Create a new node

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

new\_node->data = data;

// Set XOR of new node's address and next node's address

new\_node->xor\_ptr = \*head\_ref;

// If list is not empty, then update the previous node's XOR

if (\*head\_ref != NULL) {

// XOR of new node's address and next node's XOR

(\*head\_ref)->xor\_ptr = xor(new\_node, (\*head\_ref)->xor\_ptr);

}

// Change head

\*head\_ref = new\_node;

}

// Function to print the XOR linked list

void printList(Node\* head) {

Node \*prev = NULL, \*curr = head, \*next;

printf("XOR Linked List: ");

while (curr != NULL) {

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

// Get next node

next = xor(prev, curr->xor\_ptr);

prev = curr;

curr = next;

}

printf("\n");

}

int main() {

Node\* head = NULL;

int data, count;

printf("Enter the number of elements: ");

scanf("%d", &count);

printf("Enter %d elements:\n", count);

for (int i = 0; i < count; i++) {

scanf("%d", &data);

insert(&head, data);

}

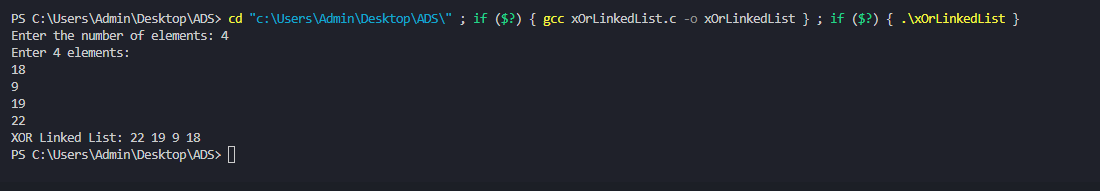
// Print the XOR linked list

printList(head);

return 0;

}

**OUTPUT:**

****