// PROBLEM STATEMENT:

// Write a C program to implement a doubly linked list using single pointer. Also

// display the linked list from:

// I. left to right

// II. right to left

// III. both sides without overlapping

// CODE:

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node \* next;

struct Node \* prev;

} Node;

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

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

new\_node -> data = data;

new\_node -> prev = NULL;

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

new\_node -> next = NULL;

\* head\_ref = new\_node;

} else {

Node \* curr = \* head\_ref;

while (curr -> next != NULL) {

curr = curr -> next;

}

curr -> next = new\_node;

new\_node -> prev = curr;

new\_node -> next = NULL;

}

}

void printLeftToRight(Node \* head) {

printf("Left to Right: ");

while (head != NULL) {

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

head = head -> next;

}

printf("\n");

}

void printRightToLeft(Node \* tail) {

printf("Right to Left: ");

while (tail != NULL) {

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

tail = tail -> prev;

}

printf("\n");

}

void printAlternate(Node \* head, Node \* tail) {

if (head == NULL) {

printf("Empty linked list!!!\n");

return;

}

printf("Alternate: ");

while (head != tail && head -> prev != tail) {

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

head = head -> next;

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

tail = tail -> prev;

}

if (head == tail) {

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

}

printf("\n\n");

}

int main() {

Node \* head = NULL;

int n;

int data;

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

scanf("%d", & n);

printf("Enter the data:\n");

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

scanf("%d", & data);

insert( & head, data);

}

printLeftToRight(head);

Node \* tail = head;

while (tail -> next != NULL) {

tail = tail -> next;

}

printRightToLeft(tail);

printAlternate(head, tail);

return 0;

}

// OUTPUT:

Enter size of linked list:

5

Enter the data:

1 2 5 8 3

Left to Right: 1 2 5 8 3

Right to Left: 3 8 5 2 1

Alternate: 1 3 2 8 5

Enter size of linked list:

6

Enter the data:

1 7 3 0 2 8

Left to Right: 1 7 3 0 2 8

Right to Left: 8 2 0 3 7 1

Alternate: 1 8 7 2 3 0

// Problem Statement:

// Write a C program to create doubly linked list by inserting nodes in such a way

// that the resultant linked list remains in ascending order.

// CODE:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \* prev;

struct Node \* next;

};

struct Node \* createNewNode(int data) {

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

newNode -> data = data;

newNode -> prev = NULL;

newNode -> next = NULL;

return newNode;

}

void insertNode(struct Node \*\* head, int data) {

struct Node \* newNode = createNewNode(data);

struct Node \* current = \* head;

// if the list is empty, make the new node as the head

if ( \* head == NULL) {

\* head = newNode;

return;

}

// if the new node's data is smaller than the head node's data,

// insert the new node before the head node

if (data < current -> data) {

newNode -> next = current;

current -> prev = newNode;

\* head = newNode;

return;

}

// find the correct position to insert the new node

while (current -> next != NULL && current -> next -> data < data) {

current = current -> next;

}

// insert the new node at the correct position

newNode -> prev = current;

newNode -> next = current -> next;

if (current -> next != NULL) {

current -> next -> prev = newNode;

}

current -> next = newNode;

}

void displayList(struct Node \* head) {

struct Node \* current = head;

while (current != NULL) {

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

current = current -> next;

}

printf("\n");

}

int main() {

struct Node \* head = NULL;

int n;

int data;

// insert nodes in ascending order

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

scanf("%d", & n);

printf("Enter the data:\n");

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

scanf("%d", & data);

insertNode( & head, data);

}

// display the linked list

printf("Doubly linked list: ");

displayList(head);

}

// OUTPUT:

Enter size of linked list:

5

Enter the data:

9 3 8 5 1

Doubly linked list: 1 3 5 8 9