Practical No:- 2 (b)

1. (b) Write a program to create double linked list and sort the elements in the linked list.

Practical Implementation:-

Code:-

// on a doubly linked list

#include <bits/stdc++.h>

using namespace std;

// Node of a doubly linked list

struct Node {

int data;

struct Node\* prev, \*next;

};

// function to create and return a new node

// of a doubly linked list

struct Node\* getNode(int data)

{

// allocate node

struct Node\* newNode =

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

// put in the data

newNode->data = data;

newNode->prev = newNode->next = NULL;

return newNode;

}

// function to insert a new node in sorted way in

// a sorted doubly linked list

void sortedInsert(struct Node\*\* head\_ref, struct Node\* newNode)

{

struct Node\* current;

// if list is empty

if (\*head\_ref == NULL)

\*head\_ref = newNode;

// if the node is to be inserted at the beginning

// of the doubly linked list

else if ((\*head\_ref)->data >= newNode->data) {

newNode->next = \*head\_ref;

newNode->next->prev = newNode;

\*head\_ref = newNode;

}

else {

current = \*head\_ref;

// locate the node after which the new node

// is to be inserted

while (current->next != NULL &&

current->next->data < newNode->data)

current = current->next;

/\*Make the appropriate links \*/

newNode->next = current->next;

// if the new node is not inserted

// at the end of the list

if (current->next != NULL)

newNode->next->prev = newNode;

current->next = newNode;

newNode->prev = current;

}

}

// function to sort a doubly linked list using insertion sort

void insertionSort(struct Node\*\* head\_ref)

{

// Initialize 'sorted' - a sorted doubly linked list

struct Node\* sorted = NULL;

// Traverse the given doubly linked list and

// insert every node to 'sorted'

struct Node\* current = \*head\_ref;

while (current != NULL) {

// Store next for next iteration

struct Node\* next = current->next;

// removing all the links so as to create 'current'

// as a new node for insertion

current->prev = current->next = NULL;

// insert current in 'sorted' doubly linked list

sortedInsert(&sorted, current);

// Update current

current = next;

}

// Update head\_ref to point to sorted doubly linked list

\*head\_ref = sorted;

}

// function to print the doubly linked list

void printList(struct Node\* head)

{

while (head != NULL) {

cout << head->data << " ";

head = head->next;

}

}

// function to insert a node at the beginning of

// the doubly linked list

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

{

/\* allocate node \*/

struct Node\* new\_node =

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

/\* put in the data \*/

new\_node->data = new\_data;

/\* Make next of new node as head and previous as NULL \*/

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

new\_node->prev = NULL;

/\* change prev of head node to new node \*/

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

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

/\* move the head to point to the new node \*/

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

}

// Driver program to test above

int main()

{

/\* start with the empty doubly linked list \*/

struct Node\* head = NULL;

// insert the following data

push(&head, 9);

push(&head, 3);

push(&head, 5);

push(&head, 10);

push(&head, 12);

push(&head, 8);

cout << "Doubly Linked List Before Sorting";

printList(head);

insertionSort(&head);

cout << "nDoubly Linked List After Sorting";

printList(head);

return 0;

}

Output:-

