Assignment # 1

Question 4

Q4:Write a method which can insert and delete an element in a sorted link list?

Solution:

#include<iostream>

using namespace std;

//structure of list

typedef struct linklist\_node {

int detail;

struct linklist\_node \*NEXT; // pointer to NEXT node in the linklist

}node;

//display list

void dsplinklist(node \*head) {

cout << "LINKLIST : ";

while (head != NULL)

{

cout << head->detail << " > ";

head = head->NEXT;

}

cout << "NULL " << endl;

}

//create new node

node \*Newnode(int detail) {

node \*Newnode = new node;

Newnode->detail = detail;

Newnode->NEXT = NULL;

return Newnode;

}

// Search the node with element as detail

node \*findnode(node \*head, int detail) {

node \*pointer = NULL;

while (head) {

if (head->detail == detail) {

pointer = head;

break;

}

head = head->NEXT;

}

return pointer;

}

// insert a node at the beginning

node \*InsrtNodeInBegining(node \*head, int detail) {

node \*pointer = Newnode(detail);

if (head == NULL) { // if list is empty

head = pointer;

}

else {

pointer->NEXT = head;

head = pointer;

}

return head;

}

// insert a node at the end

node \*InsrtNodeInEnding(node \*head, int detail) {

node \*pointer = Newnode(detail);

if (head == NULL) { //if list is empty

head = pointer;

}

else {

node \*Ntemp = head;

while (Ntemp->NEXT != NULL) { // move to the last node

Ntemp = Ntemp->NEXT;

}

Ntemp->NEXT = pointer; // insert node at the end

}

return head;

}

// insert a node at the after a particular node in the list

node \*InsrtNodeInAfter(node \*head, int element, int detail) {

// search the element after which node is to be inserted

node \*NNtemp = findnode(head, element);

if (NNtemp == NULL) { // element not found

cout << "Element not found ... " << endl;

}

else {

node \*pointer = Newnode(detail);

if (NNtemp->NEXT == NULL) { // node has to inserted after the last node

NNtemp->NEXT = pointer;

}

else { // insert the node after the first or an intermediate node

pointer->NEXT = NNtemp->NEXT;

NNtemp->NEXT = pointer;

}

}

return head;

}

/\* delete a particular node from the list \*/

node \*delnode(node \*head, int element) {

node \*NNtemp = findnode(head, element); // search the node to be deleted

if (NNtemp == NULL) { // element not found

cout << "node not found ... " << endl;

}

else {

if (NNtemp == head) { // first node is to be deleted

head = head->NEXT;

delete NNtemp;

}

else { // node to deleted is an intermediate or last node

node \*pointer = head;

while (pointer->NEXT != NNtemp) {

pointer = pointer->NEXT;

}

pointer->NEXT = NNtemp->NEXT;

delete NNtemp;

}

}

return head;

}

int main() {

node \*head = NULL;

head = InsrtNodeInBegining(head, 104); // 104

head = InsrtNodeInBegining(head, 100); // 100 -> 104

head = InsrtNodeInEnding(head, 108); // 100 -> 104 -> 108

head = InsrtNodeInAfter(head, 100, 102); // 100 -> 102 -> 104 -> 108

head = InsrtNodeInAfter(head, 104, 106); // 100 -> 102 -> 104 -> 106 -> 108

head = InsrtNodeInAfter(head, 108, 110); // 100 -> 102 -> 104 -> 106 -> 108 -> 110

dsplinklist(head);

head = delnode(head, 104); // 100 -> 102 -> 106 -> 108 -> 110 after deleting 104

head = delnode(head, 110); // 100 -> 102 -> 106 -> 108 after deleting 110

head = delnode(head, 100); // 102 -> 106 -> 108 after deleting 100

dsplinklist(head);

return 0;

}