**VARUN KUMAR**

**2K19-IT-140**

**DS LAB - 8**

15: Given a Sorted doubly linked list of positive integers and an integer, then finds all the pairs (sum of two nodes data part) that is equal to the given integer value. Example: Double Linked List 2, 5, 7, 8, 9, 10, 12, 16, 19, 25, and P=35 then pairs will be Pairs will be (10, 25), (16, 19).

#include<bits/stdc++.h>

using namespace std;

class ListNode

{ public:

int val;

class ListNode \*next, \*prev;

};

void Pair\_Sum(class ListNode \*head, int p)

{

class ListNode \*first = head;

class ListNode \*second = head;

while (second->next != NULL)

second = second->next;

bool found = false;

while (first != NULL && second != NULL &&

first != second && second->next != first)

{

if ((first->val + second->val) == p)

{

found = true;

cout << "(" << first->val<< ", "

<< second->val << ")" << endl;

first = first->next;

second = second->prev;

}

else

{

if ((first->val + second->val) < p)

first = first->next;

else

second = second->prev;

}

}

if (found == false)

cout << "No pair found";

}

void insert\_in\_list(class ListNode \*\*head, int val)

{

class ListNode \*temp = new ListNode;

temp->val = val;

temp->next = temp->prev = NULL;

if (!(\*head))

(\*head) = temp;

else

{

temp->next = \*head;

(\*head)->prev = temp;

(\*head) = temp;

}

}

int main()

{

class ListNode \*head = NULL;

int n;

cout<<"ENTER NO OF ELEMENTS : "; cin>>n;

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

int a; cin>>a;

insert\_in\_list(&head,a);

}

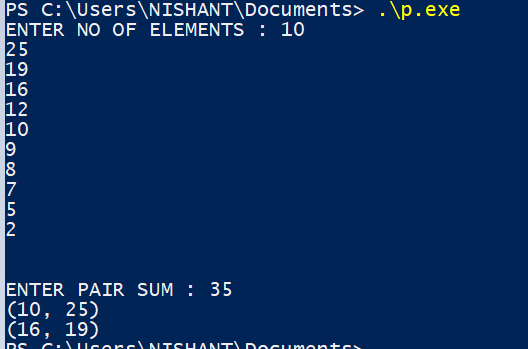
int p;

cout<<"\n\nENTER PAIR SUM : "; cin>>p;

Pair\_Sum(head,p );

return 0;

}



# 16: Write a program to implement the Binary Tree using linked list and perform In-order traversal.

#include <iostream>

#include <string>

#include <queue>

using namespace std;

class ListNode

{ public:

int data;

ListNode\* next;

};

class TreeNode

{ public:

int data;

TreeNode \*left, \*right;

};

void insert(class ListNode\*\* head\_ref, int new\_data)

{

class ListNode\* new\_node = new ListNode;

new\_node->data = new\_data;

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

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

}

TreeNode\* New\_Tree(int data)

{

TreeNode \*temp = new TreeNode;

temp->data = data;

temp->left = temp->right = NULL;

return temp;

}

void Convert\_List\_to\_binary(ListNode \*head, TreeNode\* &root)

{

queue<TreeNode \*> q;

if (head == NULL)

{

root = NULL;

return;

}

root = New\_Tree(head->data);

q.push(root);

head = head->next;

while (head)

{

TreeNode\* parent = q.front();

q.pop();

TreeNode \*leftChild = NULL, \*rightChild = NULL;

leftChild = New\_Tree(head->data);

q.push(leftChild);

head = head->next;

if (head)

{

rightChild = New\_Tree(head->data);

q.push(rightChild);

head = head->next;

}

parent->left = leftChild;

parent->right = rightChild;

}

}

void inorder\_Traversal(TreeNode\* root)

{

if (root)

{

inorder\_Traversal( root->left );

cout << root->data << " ";

inorder\_Traversal( root->right );

}

}

int main()

{

class ListNode\* head = NULL;

insert(&head, 5);

insert(&head, 13);

insert(&head, 64);

insert(&head, 100);

insert(&head, 225);

insert(&head, 4);

TreeNode \*root;

Convert\_List\_to\_binary(head, root);

cout << "\n\n\t\tInorder Traversal of the constructed Binary Tree is: \n\n\t\t";

inorder\_Traversal(root);

cout<<"\n\n";

return 0;

}

