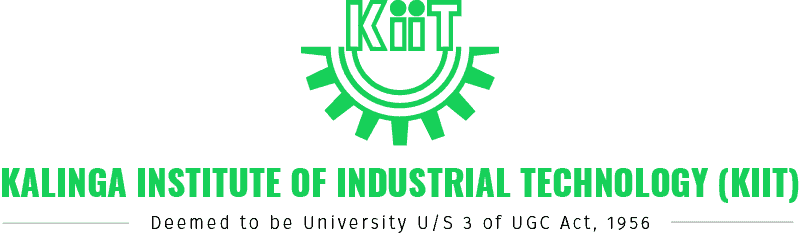
****

**NAME : SOHAM SAMANTA**

**ROLL NUMBER : 20051107**

**SECTION : CSE-14**

**DAA LAB ASSIGNMENT - 5**

**1· How do you find the middle element of a singly linked list in one pass?**

**Solution:**

//SOHAM SAMANTA CODES

#include<bits/stdc++.h>

using namespace std;

#define ll long long int

#define mod 1000000007

#define PI 3.1415926535897932384626433832

#define ss ios\_base::sync\_with\_stdio(false);cin.tie(NULL);

struct Node {

int data;

Node \*next = NULL;

};

typedef struct Node Node;

Node \*head = NULL;

void push(int value) {

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

new\_node->data = value;

if (head == NULL) {

head = new\_node;

return;

} else {

Node \*ptr;

ptr = head;

while (ptr->next != NULL) {

ptr = ptr->next;

}

ptr->next = new\_node;

}

}

int findMiddleNode() {

Node \*fast, \*slow;

fast = head;

slow = head;

while(fast != NULL && slow != NULL && fast->next->next != NULL){

fast = fast->next->next;

slow = slow->next;

}

return slow->data;

}

void display(){

Node \*temp;

temp = head;

while(temp != NULL){

cout << temp->data << "->";

temp = temp->next;

}

cout << "NULL" << endl;

}

int\_fast32\_t main(){

ss;

for(int i = 0; i < 6; i++){

int val = rand() % 30;

push(val);

}

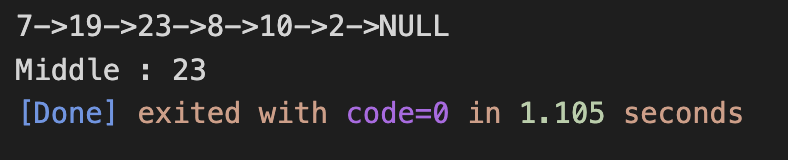
display();

cout << "Middle : " << findMiddleNode();

return 0;

}

**Output:**

****

**2· How do you check if a given linked list contains a cycle? How do you find the starting node of**

**the cycle?**

**Solution:**

//SOHAM SAMANTA CODES

#include<bits/stdc++.h>

using namespace std;

#define ll long long int

#define mod 1000000007

#define PI 3.1415926535897932384626433832

#define ss ios\_base::sync\_with\_stdio(false);cin.tie(NULL);

struct Node {

int data;

Node \*next = NULL;

};

typedef struct Node Node;

Node \*head = NULL;

void push(int val) {

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

new\_node->data = val;

if (head == NULL) {

head = new\_node;

return;

} else {

Node \*ptr;

ptr = head;

while (ptr->next != NULL) {

ptr = ptr->next;

}

ptr->next = new\_node;

}

}

bool containsCycle() {

Node \*fast, \*slow;

fast = head;

slow = head;

while (fast != NULL && slow != NULL && fast->next->next != NULL) {

fast = fast->next->next;

slow = slow->next;

if (slow == fast) {

return true;

}

}

return false;

}

void display() {

Node \*temp;

temp = head;

while (temp != NULL) {

cout << temp->data << "->";

temp = temp->next;

}

cout << "NULL" << endl;

}

int32\_t main() {

ss;

for (int i = 0; i < 6; i++) {

int val = rand() % 50;

push(val);

}

Node \*temp1;

temp1 = head;

// uncommment to check the cycle

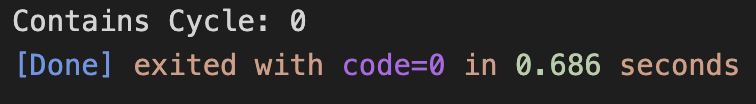
// temp1->next->next->next->next->next->next = temp1->next->next->next;

cout << "Contains Cycle: " << containsCycle();

return 0;

}

**Output:**

****

**3· How do you find the third node from the end in a singly linked list?**

**Solution:**

//SOHAM SAMANTA CODES

#include<bits/stdc++.h>

using namespace std;

#define ll long long int

#define mod 1000000007

#define PI 3.1415926535897932384626433832

#define ss ios\_base::sync\_with\_stdio(false);cin.tie(NULL);

struct Node {

int data;

struct Node\* next;

};

void print3rdFromLast(struct Node\* head)

{

int N=3;

int len = 0, i;

struct Node\* temp = head;

while (temp != NULL) {

temp = temp->next;

len++;

}

if (len < N)

return;

temp = head;

for (i = 1; i < len - N + 1; i++)

temp = temp->next;

cout << temp->data;

return;

}

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

struct Node\* new\_node = new Node();

new\_node->data = new\_data;

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

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

}

int32\_t main(){

ss;

struct Node\* head = NULL;

// 8->21->35->15->4->NULL

push(&head, 4);

push(&head, 15);

push(&head, 35);

push(&head, 21);

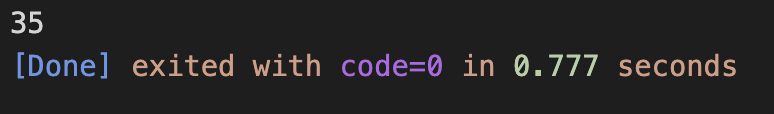
push(&head, 8);

print3rdFromLast(head);

return 0;

}

**Output:**

****

**4· How to convert a sorted list to a binary search tree?**

**Solution:**

//SOHAM SAMANTA CODES

#include<bits/stdc++.h>

using namespace std;

#define ll long long int

#define mod 1000000007

#define PI 3.1415926535897932384626433832

#define ss ios\_base::sync\_with\_stdio(false);cin.tie(NULL);

class LNode

{

public:

int data;

LNode\* next;

};

class TNode

{

public:

int data;

TNode\* left;

TNode\* right;

};

TNode\* newNode(int data);

int countLNodes(LNode \*head);

TNode\* sortedListToBSTRecur(LNode \*\*head\_ref, int n);

TNode\* sortedListToBST(LNode \*head)

{

int n = countLNodes(head);

return sortedListToBSTRecur(&head, n);

}

TNode\* sortedListToBSTRecur(LNode \*\*head\_ref, int n)

{

if (n <= 0)

return NULL;

TNode \*left = sortedListToBSTRecur(head\_ref, n/2);

TNode \*root = newNode((\*head\_ref)->data);

root->left = left;

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

root->right = sortedListToBSTRecur(head\_ref, n - n / 2 - 1);

return root;

}

int countLNodes(LNode \*head)

{

int count = 0;

LNode \*temp = head;

while(temp)

{

temp = temp->next;

count++;

}

return count;

}

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

{

LNode\* new\_node = new LNode();

new\_node->data = new\_data;

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

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

}

void printList(LNode \*node)

{

while(node!=NULL)

{

cout << node->data << " ";

node = node->next;

}

}

TNode\* newNode(int data)

{

TNode\* node = new TNode();

node->data = data;

node->left = NULL;

node->right = NULL;

return node;

}

void preOrder(TNode\* node)

{

if (node == NULL)

return;

cout<<node->data<<" ";

preOrder(node->left);

preOrder(node->right);

}

int32\_t main()

{

ss;

LNode\* head = NULL;

push(&head, 7);

push(&head, 6);

push(&head, 5);

push(&head, 4);

push(&head, 3);

push(&head, 2);

push(&head, 1);

cout<<"Given Linked List ";

printList(head);

TNode \*root = sortedListToBST(head);

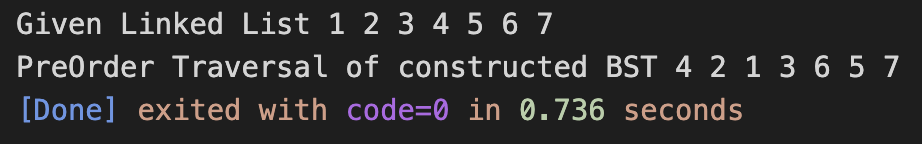
cout<<"\nPreOrder Traversal of constructed BST ";

preOrder(root);

return 0;

}

**Output:**

****

**5· Given a linked list and a value x, partition it such that all nodes less than x come before nodes**

**greater than or equal to x.**

**Solution:**

//SOHAM SAMANTA CODES

#include<bits/stdc++.h>

using namespace std;

#define ll long long int

#define mod 1000000007

#define PI 3.1415926535897932384626433832

#define ss ios\_base::sync\_with\_stdio(false);cin.tie(NULL);

struct Node {

int data;

struct Node\* next;

};

Node\* newNode(int data)

{

struct Node\* new\_node = new Node;

new\_node->data = data;

new\_node->next = NULL;

return new\_node;

}

struct Node\* partition(struct Node\* head, int x)

{

struct Node \*smallerHead = NULL, \*smallerLast = NULL;

struct Node \*greaterLast = NULL, \*greaterHead = NULL;

struct Node \*equalHead = NULL, \*equalLast = NULL;

while (head != NULL) {

if (head->data == x) {

if (equalHead == NULL)

equalHead = equalLast = head;

else {

equalLast->next = head;

equalLast = equalLast->next;

}

}

else if (head->data < x) {

if (smallerHead == NULL)

smallerLast = smallerHead = head;

else {

smallerLast->next = head;

smallerLast = head;

}

}

else

{

if (greaterHead == NULL)

greaterLast = greaterHead = head;

else {

greaterLast->next = head;

greaterLast = head;

}

}

head = head->next;

}

if (greaterLast != NULL)

greaterLast->next = NULL;

if (smallerHead == NULL) {

if (equalHead == NULL)

return greaterHead;

equalLast->next = greaterHead;

return equalHead;

}

if (equalHead == NULL) {

smallerLast->next = greaterHead;

return smallerHead;

}

smallerLast->next = equalHead;

equalLast->next = greaterHead;

return smallerHead;

}

void printList(struct Node\* head)

{

struct Node\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

}

int main()

{

struct Node\* head = newNode(10);

head->next = newNode(4);

head->next->next = newNode(20);

head->next->next->next = newNode(10);

head->next->next->next->next = newNode(3);

// head->next->next->next->next->next = newNode(4);

cout<<"Before Partition: "<<endl;

printList(head);

cout<<"\nAfter Partition: "<<endl;

int x = 3;

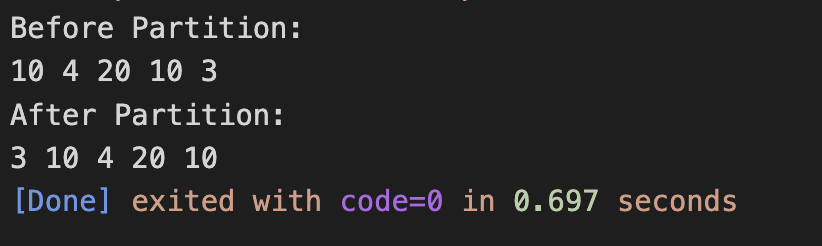
head = partition(head, x);

printList(head);

return 0;

}

**Output:**

****

**6· How do you reverse a singly linked list without recursion?**

**Solution:**

//SOHAM SAMANTA CODES

#include<bits/stdc++.h>

using namespace std;

// #define ll long long int

#define mod 1000000007

#define PI 3.1415926535897932384626433832

#define ss ios\_base::sync\_with\_stdio(false);cin.tie(NULL);

#include <iostream>

using namespace std;

struct Node {

int data;

struct Node\* next;

Node(int data)

{

this->data = data;

next = NULL;

}

};

struct LinkedList {

Node\* head;

LinkedList() { head = NULL; }

void reverse()

{

Node\* current = head;

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

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

void print()

{

struct Node\* temp = head;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

}

void push(int data)

{

Node\* temp = new Node(data);

temp->next = head;

head = temp;

}

};

int main()

{

LinkedList ll;

ll.push(20);

ll.push(4);

ll.push(15);

ll.push(85);

cout << "Given linked list\n";

ll.print();

ll.reverse();

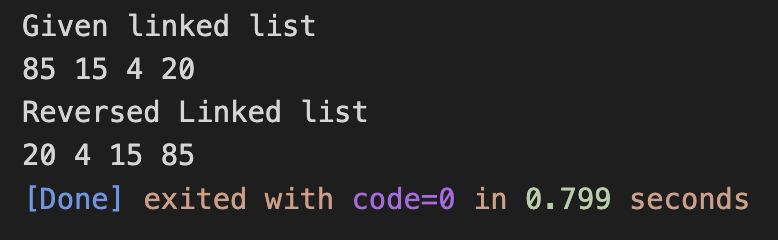
cout << "\nReversed Linked list \n";

ll.print();

return 0;

}

**Output:**

****

**7· How are duplicate nodes removed in an unsorted linked list?**

**Solution:**

//SOHAM SAMANTA CODES

#include<bits/stdc++.h>

using namespace std;

#define ll long long int

#define mod 1000000007

#define PI 3.1415926535897932384626433832

#define ss ios\_base::sync\_with\_stdio(false);cin.tie(NULL);

#include <bits/stdc++.h>

using namespace std;

struct Node {

int data;

struct Node\* next;

};

struct Node\* newNode(int data)

{

Node\* temp = new Node;

temp->data = data;

temp->next = NULL;

return temp;

}

void removeDuplicates(struct Node\* start)

{

struct Node \*ptr1, \*ptr2, \*dup;

ptr1 = start;

while (ptr1 != NULL && ptr1->next != NULL) {

ptr2 = ptr1;

while (ptr2->next != NULL) {

if (ptr1->data == ptr2->next->data) {

dup = ptr2->next;

ptr2->next = ptr2->next->next;

delete (dup);

}else

ptr2 = ptr2->next;

}

ptr1 = ptr1->next;

}

}

void printList(struct Node\* node)

{

while (node != NULL) {

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

node = node->next;

}

}

int32\_t main()

{

ss;

struct Node\* start = newNode(10);

start->next = newNode(12);

start->next->next = newNode(11);

start->next->next->next = newNode(11);

start->next->next->next->next = newNode(12);

start->next->next->next->next->next = newNode(11);

start->next->next->next->next->next->next = newNode(10);

printf("Linked list before removing duplicates ");

printList(start);

removeDuplicates(start);

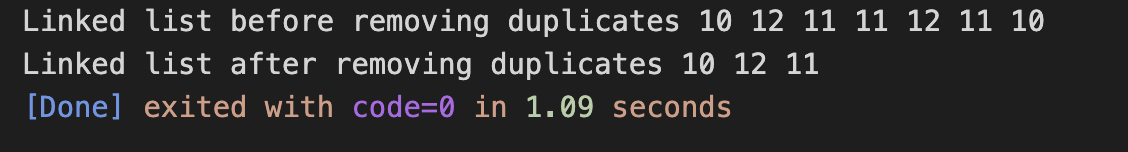
printf("\nLinked list after removing duplicates ");

printList(start);

return 0;

}

**Output:**

****

