Data Structures Lab #6

Name: Ahmed Kasteer

Roll Number: 20F-0336

**Question # 1**

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* link;

};

Node\* head = NULL;

void InsertAtBeg(Node\* &head,int val)

{

Node\* node = new Node();

node->data = val;

node->link = head;

head = node;

}

void insertAtMid(Node\* &head, int x)

{

Node\* newNode = new Node();

if (head == NULL)

{

head = newNode;

}

else

{

Node\* newNode = new Node();

newNode->data = x;

Node\* ptr = head;

int length = 0;

while (ptr != NULL)

{

length++;

ptr = ptr->link;

}

int count;

if (length % 2 == 0)

{

count = length / 2;

}

else

{

count = (length + 1) / 2;

}

ptr = head;

while (count-- > 1)

{

ptr = ptr->link;

}

newNode->link = ptr->link;

ptr->link = newNode;

}

}

void insertAtEnd(Node\* &head,int val)

{

Node \*node = new Node();

node->data = val;

node->link = NULL;

if (head == NULL)

{

head = node;

}

else

{

Node\* temp = new Node();

temp = head;

while (temp->link != NULL)

{

temp = temp->link;

}

temp->link = node;

}

}

void Display(Node\* head)

{

Node\* temp = head;

while (temp != NULL)

{

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

temp = temp->link;

}

cout << "NULL" << endl;

}

int main()

{

Node\* head = NULL;

insertAtEnd(head, 6);

insertAtEnd(head, 4);

insertAtEnd(head, 22);

insertAtEnd(head,40);

insertAtEnd(head, 100);

Display(head);

InsertAtBeg(head, 30);

Display(head);

insertAtMid(head, 15);

Display(head);

}

**A screenshot of a computer screen

Description automatically generated with medium confidence**

**Question #2**

#include <iostream>

using namespace std;

struct node

{

int data;

node\* next;

};

class LinkedList

{

public:

node\* head;

LinkedList()

{

head = NULL;

}

void CreateNode(int n)

{

node\* temp = new node();

node\* newNode = new node();

newNode->data = n;

newNode->next = NULL;

if (head == NULL)

{

head = newNode;

}

else

{

temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = newNode;

}

}

void display(node\* head)

{

node\* temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

cout << "NULL" << endl;

}

void merge(node\* &head1, node\* &head2)

{

node\* temp = head1;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = head2;

}

};

int main()

{

LinkedList a;

a.CreateNode(5);

a.CreateNode(500);

a.CreateNode(145);

cout << "Linked List 1: ";

a.display(a.head);

LinkedList b;

b.CreateNode(3);

b.CreateNode(300);

b.CreateNode(7);

cout << "Linked List 2: ";

b.display(b.head);

a.merge(a.head, b.head);

cout << "Linked List Merge: ";

a.display(a.head);

return 0;

}

**Text

Description automatically generated**

**Question #4**

#include <iostream>

using namespace std;

struct node

{

int data;

node\* next;

};

class LL

{

node\* head = new node();

node\* tail = new node();

public:

int CreateNode(int n)

{

int sum = 0;

node\* temp = new node;

temp->data = n;

temp->next = NULL;

if (head == NULL)

{

head = temp;

tail = temp;

}

else

{

node\* newnode = new node();

node\* temp = new node();

temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = newnode;

}

sum = sum + temp->data;

return sum;

}

};

int main()

{

int s = 0;

LL a;

for (int i = 0; i < 15; i++)

{

s += a.CreateNode(rand() % 100);

}

cout << "Sum of nodes: " << s << endl;

cout << "Average: " << s / 10 << endl;

}

**A picture containing text

Description automatically generated**

**Question #5**

#include<iostream>

using namespace std;

class node

{

public:

int data;

node\* next;

node\* prev;

node(int val)

{

data = val;

next = NULL;

prev = NULL;

}

};

void insertathead(node\*& head, int val)

{

node\* n = new node(val);

n->next = head;

if (head != NULL)

{

head->prev = n;

}

head = n;

}

void insertatTail(node\*& head, int val)

{

if (head == NULL)

{

insertathead(head, val);

return;

}

node\* n = new node(val);

node\* temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = n;

n->prev = temp;

}

void display(node\* head)

{

node\* temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

cout << "NULL" << endl;

}

void reverse(node\*& head)

{

node\* temp = NULL;

node\* current = head;

while (current != NULL)

{

temp = current->prev;

current->prev = current->next;

current->next = temp;

current = current->prev;

}

if (temp != NULL)

head = temp->prev;

}

int main()

{

node\* head = NULL;

insertatTail(head, 1);

insertatTail(head, 2);

insertatTail(head, 3);

insertatTail(head, 4);

insertatTail(head, 5);

insertatTail(head, 6);

insertatTail(head, 7);

insertatTail(head, 8);

insertatTail(head, 9);

insertatTail(head, 10);

cout << "Linked list before reverse: ";

display(head);

cout << endl;

cout << "Linked list after reverse: ";

reverse(head);

display(head);

}

**Text

Description automatically generated**

**Question #6**

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* link;

};

Node\* head = NULL;

void InsertAtBeg(Node\*& head, int val)

{

Node\* node = new Node();

node->data = val;

node->link = head;

head = node;

}

void insertAtMid(Node\*& head, int x)

{

Node\* newNode = new Node();

if (head == NULL)

{

head = newNode;

}

else

{

Node\* newNode = new Node();

newNode->data = x;

Node\* ptr = head;

int length = 0;

while (ptr != NULL)

{

length++;

ptr = ptr->link;

}

int count;

if (length % 2 == 0)

{

count = length / 2;

}

else

{

count = (length + 1) / 2;

}

ptr = head;

while (count-- > 1)

{

ptr = ptr->link;

}

newNode->link = ptr->link;

ptr->link = newNode;

}

}

void insertAtEnd(Node\*& head, int val)

{

Node\* node = new Node();

node->data = val;

node->link = NULL;

if (head == NULL)

{

head = node;

}

else

{

Node\* temp = new Node();

temp = head;

while (temp->link != NULL)

{

temp = temp->link;

}

temp->link = node;

}

}

void Display(Node\* head)

{

Node\* temp = head;

while (temp != NULL)

{

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

temp = temp->link;

}

cout << "NULL" << endl;

}

int main()

{

Node\* head = NULL;

insertAtEnd(head, 6);

insertAtEnd(head, 4);

insertAtEnd(head, 22);

insertAtEnd(head, 40);

insertAtEnd(head, 100);

Display(head);

InsertAtBeg(head, 30);

Display(head);

cout << "Inserting at middle" << endl;

insertAtMid(head, 15);

Display(head);

}

**Text

Description automatically generated**

**Question #7**

#include <iostream>

using namespace std;

class Node

{

public:

int data;

Node\* next;

};

void moveToFront(Node\*\* headptr)

{

if (\*headptr == NULL || (\*headptr)->next == NULL)

return;

Node\* secondlast = NULL;

Node\* last = \*headptr;

while (last->next != NULL)

{

secondlast = last;

last = last->next;

}

secondlast->next = NULL;

last->next = \*headptr;

\*headptr = last;

}

void insertNode(Node\*\* headptr, int value)

{

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

new\_node->data = value;

new\_node->next = (\*headptr);

(\*headptr) = new\_node;

}

void DisplayList(Node\* node)

{

while (node != NULL)

{

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

node = node->next;

}

cout << "NULL" << endl;

}

int main()

{

Node\* start = NULL;

insertNode(&start, 4);

insertNode(&start, 3);

insertNode(&start, 2);

insertNode(&start, 1);

cout << "List Before moving last node to front: ";

DisplayList(start);

cout << endl << endl;

moveToFront(&start);

cout << "List after removing last node to front: ";

DisplayList(start);

cout << endl;

}

**Text

Description automatically generated**

**Question #9**

#include<iostream>

using namespace std;

class node

{

public:

int data;

node\* next;

node\* prev;

node(int val)

{

data = val;

next = NULL;

prev = NULL;

}

};

void insertathead(node\*& head, int val)

{

node\* n = new node(val);

n->next = head;

if (head != NULL)

{

head->prev = n;

}

head = n;

}

void insertatTail(node\*& head, int val)

{

if (head == NULL)

{

insertathead(head, val);

return;

}

node\* n = new node(val);

node\* temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = n;

n->prev = temp;

}

void display(node\* head)

{

node\* temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

cout << "NULL" << endl;

}

void deleteNode(node\* head, node\* del)

{

if (head == NULL || del == NULL)

return;

if (head == del)

head = del->next;

if (del->next != NULL)

del->next->prev = del->prev;

if (del->prev != NULL)

del->prev->next = del->next;

free(del);

}

void removeDuplicates(node\* head)

{

if (head == NULL ||

head->next == NULL)

return;

node\* temp1, \* temp2, \* next;

for (temp1 = head; temp1 != NULL; temp1 = temp1->next)

{

temp2 = temp1->next;

while (temp2 != NULL) {

if (temp1->data == temp2->data)

{

next = temp2->next;

deleteNode(head, temp2);

temp2 = next;

}

else

temp2 = temp2->next;

}

}

}

int main()

{

node\* head = NULL;

insertatTail(head, 1);

insertatTail(head, 1);

insertatTail(head, 2);

insertatTail(head, 3);

insertatTail(head, 3);

cout << "linked list: ";

display(head);

removeDuplicates(head);

cout << endl;

cout << "After removing duplicates: ";

display(head);

}

**A screenshot of a computer screen

Description automatically generated with medium confidence**

**Question #10**

#include <iostream>

using namespace std;

struct linkedList

{

int number;

linkedList\* next;

};

void AddNode()

{

linkedList\* head = NULL;

linkedList\* current ;

for (int i = 0; i < 10; i++)

{

current = new linkedList;

current->number = i;

current->next = head;

head = current;

}

while (current->next != NULL)

{

cout << current->number << "->" ;

current = current->next;

}

cout << "NULL" << endl;

head = current;

delete current;

}

int main()

{

AddNode();

}

Text

Description automatically generated

**Question #11**

#include <iostream>

using namespace std;

struct myList {

int data;

myList\* next;

};

int main() {

myList\* head;

myList\* cur;

myList\* previous = NULL;

for (int i = 0; i < 4; i++) {

head = new myList;

head->data = 0;

head->next = previous;

for (cur = previous; cur != NULL; cur = cur->next)

head->data += 1 + 2 \* cur->data;

previous = head;

}

while (previous != NULL) {

cout << previous->data << endl;

cur = previous;

previous = previous->next;

delete cur;

}

return 0;

}

**Text

Description automatically generated**