**DATA STRUCTURES LAB 5**

Name: Ahmed Kasteer

Roll no: 20F0336

Section 3D

Question 2

#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;

}

}

void sort(node\* head)

{

int swapped, i;

node\* temp;

node\* temp2 = NULL;

if (head == NULL)

{

return;

}

do

{

swapped = 0;

temp = head;

while (temp->next != temp2)

{

if (temp->data > temp->next->data)

{

swap(temp->data, temp->next->data);

swapped = 1;

}

temp = temp->next;

}

temp2 = temp;

} while (swapped);

}

int main()

{

node\* head = NULL;

insertatTail(head, 11);

insertatTail(head, 21);

insertatTail(head, 4);

insertatTail(head, 2);

insertatTail(head, 5);

cout << "Doubly linked list: ";

display(head);

sort(head);

cout << endl;

cout << "After sorting: ";

display(head);

}

Text

Description automatically generated

Question 3

#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;

}

}

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 << "Doubly linked list: ";

display(head);

removeDuplicates(head);

cout << endl;

cout << "After removing duplicates: ";

display(head);

}

Text

Description automatically generated

Question 4

#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;

}

}

void storingEvenAndOdd(node\*& head1, node\*& head2, node\*& head)

{

node\* temp = head;

while (temp != NULL)

{

if (temp->data % 2 == 0)

{

insertatTail(head1, temp->data);

}

else

{

insertatTail(head2, temp->data);

}

temp = temp->next;

}

}

int main()

{

node\* head = NULL;

node\* even = NULL;

node\* odd = NULL;

insertatTail(head, 1);

insertatTail(head, 1);

insertatTail(head, 2);

insertatTail(head, 3);

insertatTail(head, 4);

insertatTail(head, 5);

insertatTail(head, 5);

insertatTail(head, 6);

insertatTail(head, 7);

insertatTail(head, 8);

storingEvenAndOdd(even, odd, head);

cout << "Linked list: ";

display(head);

cout << endl;

cout << "Even linked list: ";

display(even);

cout << endl;

cout << "Odd linked list: ";

display(odd);

}

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;

}

}

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 7

#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)

{

if (temp->next != head)//if I remove this condition then the circular linked list will print in an endless loop

{

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

temp = temp->next;

}

}

}

void circular(node\*& head)

{

node\* temp = head;

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = head;

}

int main()

{

node\* head = NULL;

insertatTail(head, 1);

insertatTail(head, 2);

insertatTail(head, 3);

insertatTail(head, 4);

insertatTail(head, 5);

cout << "Linked list before circular: ";

display(head);

cout << endl;

cout << "Linked list after circular: ";

circular(head);

display(head);

}

Text

Description automatically generated with medium confidence

Question 8

#include<iostream>

using namespace std;

class node

{

public:

int data;

node\* next;

node(int val)

{

data = val;

next = NULL;

}

};

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

{

node\* n = new node(val);

head = n;

n->next = head;

}

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

{

if (head == NULL)

{

insert(head, val);

return;

}

node\* n = new node(val);

n->next = head->next;

head->next = n;

}

void deleteAtEnd(node\*& head)

{

node\* temp = head;

node\* prev = NULL;

if (head == NULL)

{

cout << "List is empty" << endl;

return;

}

if (temp->next == temp)

{

head = NULL;

return;

}

while (temp->next != head)

{

prev = temp;

temp = temp->next;

}

prev->next = temp->next;

head = prev->next;

free(temp);

}

void display(node\* head)

{

node\* ptr;

ptr = head;

do

{

cout << ptr->data << " ";

ptr = ptr->next;

} while (ptr != head);

}

int main()

{

node\* head = NULL;

insertatBegin(head, 5);

insertatBegin(head, 4);

insertatBegin(head, 3);

insertatBegin(head, 2);

insertatBegin(head, 1);

cout << "Circular Linked list: ";

display(head);

cout << endl;

cout << "After deleting last node: ";

deleteAtEnd(head);

display(head);

}

Text

Description automatically generated