Hassan Raza
BAI
2024229

CS 221 Assignment 2

Question: 1.

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node *next; };

void detect_and_remove_loop (Node* head) {
    Node* slow = head;
    Node* fast = head;
    bool loopexists = false;

    while (fast && fast->next) {
      slow = slow->next;
      fast = fast->next->next;
      if (slow == fast) {
        loopexists = true;
      } break;

    }

    if (!loopexists) {
        cout << "No loop detected!" << endl;
        return; }

  slow = head
  int pos=1
  while ( slow != fast) {
      slow = slow->next;
    fast = fast->next;
    pos++; }
cout << "Loop starts at Node # " << pos << endl;
cout << "value at Node = " << slow->data << endl;
```

```
Node* loopstart = slow;
Node* temp = loopstart;
while (temp->next != loopstart) {
    temp = temp->next; }

cout<< "loop removed from Node # "<<pos;
   << " <- Node with value "<<temp->data<<endl
temp->next = Null;
}

void printlist(Node *head) {
    Node* temp = head;
    while(temp) {
        cout<< temp->data;
        cout<<endl; }
}

int main() {
    Node* head = new Node(); Node* second = new Node(); Node* third = new Node();
    head->data = 11;
    head->next =
    second->data = 3;
    third data = 8;
    head->next = second;
    second->next = third;
    third->next = second;
    detect_and_remove_loop(head);
    printlist(head);
    return 0;
}
```

Question : 2-

```
#includ <iostream>
#include <string>
using namespace std;

class Person {
  Public:
    string name;
    string address;
    string phone;
```

```
.son() {}
erson (string n, string a, string p) {
    name = n;
    address = a;
    phone = p; }
};

class Node{
  Public:
    Person data;
    Node* next;
    Node (Person p) {
        data = p;
};      next = NULL; }

class AddressBook() {
    Node* head;
 Public:
    AddressBook() {
        head = NULL;
    }

 void insert(Person p) {
    Node* newNode = new Node(p);
    if(!head || p.name < head->data.name) {
        newnode -> next = head;
        head = newNode;
        return;
      }
   Node* temp = head;
   while(temp -> next && temp->next -> data.name < p.name) {
        temp = temp -> next;
    }
   newNode -> next = temp -> next
   temp -> next = newNode
}

void Find (string name) {
    Node* temp = head;
    while (temp) {
        if(temp ->data.name == name) {
      cout << "Record found: \n";
        return;
    }
    temp = temp -> next;
}
cout << "Record not found " << name << endl;
```

```
void display() {
    Node* temp = head;
    cout << "\n Address Book Records: \n";
    while (temp) {
        cout << temp->data.name << ", "
             << temp->data.address << ", "
             << temp->data.phone << endl;
        temp = temp->next;
    }
};

int main()
{
Address Book book;
  int n;
  cout << "Enter number of records: ";
  cin >> n;
  cin.ignore();

  for (int i=0; i<n; i++) {
    string name, address, phone;
    cout << "Enter values: ";
    getline(cin, name);
    getline(cin, address);
    getline(cin, phone)

    person p(name, address, phone);
    book.insert(p);
  }

  book.display();

  string searchName;
  cout << "\n Enter name to search: ";
  getline(cin, searchname);
  book.Find(searchname);

  return 0;
```

ion: 3 -

```
include <iostream>
using namespace std;
truct Node {
 int data;
 Node *prev;
 Node * next;
  Node(int val) : data(val), prev(nullptr), next(nullptr) {}
};
truct DoublyLinkedList {
  Node* head;
  Node * tail;
  DoublyLinkedList () : head(nullptr), tail(nullptr) {}
  void append(int val) {
      Node* newNode = new Node (val);
          if (head == nullptr) {
             head = newNode;
             tail = newNode;
            }
            else {
              tail -> next = newNole
              : newNode -> previous = tail;
                tail = new Node ;
                }
        }
  Node* FindNode(int val) {
       Node* current = head;
         while ( current != nullptr) {
              if (current -> data == val) {
                    return current;
               }
             current = current -> next;
          }
         return nullptr;
       }
void  PrintList () {
      Node* current = head;
       while (current != nullptr) {
           cout << current -> data << " <-> ";
             current = current -> next;
       }
  cout << "nullptr" << endl;
}
```

```
void swapnodes (Node* a, Node *b) {
    if (a==nullptr || b==nullptr || a==b) {
        cout << " Invalid nodes for swapping." << endl;
        return;
    }

cout << " swapping nodes with values " << a->data << "and" << b->data;
    if (a->next ==b) {
        a->next = b->next;
        if (b->next)
            b->next->prev =a;
        b->prev = a->prev;
        if (a->prev)
            a->prev->next=b;
        b->next =a;
        a->prev = b;
        if(head ==a)
            head = b;
        if (tail ==b)
            tail = a;
        return;
    }
else if (b->next == a) {
        swapNodes(b,a);
        return;
    }
Node* a_prev = a->prev;
Node* a_next = a->next;
Node* b_prev = b->prev;
Node b_next = b->next;

if (a_prev)
    a_prev->next =b;
if (a_next)
    a_next->prev =b;
if (b_prev)
    b_prev->next =a;
if (b_next)
    b_next->prev =a;
```

string address;
string phone;

```
(a->prev, b->prev);
wap(a->next, b->next);

if(head==a)
    head=b;
else if(head==b)
    head=a;

if(tail==a)
    tail=b;
else if(tail==b)
    tail=a;
}
};

int main() {
    DoublyLinkedList mylist;
    mylist.append(3);
    mylist.append(1);
    mylist.append(8);
    mylist.append(6);
    mylist.append(0);

    cout<<"original list:";
    mylist.printlist();

    Node* nodeA = mylist.findNode(8);
    Node* nodeB = mylist.findNode(3);
    mylist.swapNodes(nodeA, nodeB);
    cout<<"list after swapping 8 and 6:";
    mylist.printlist();

    return 0;
}
```

Question 4:-

```
#include <iostream>
using namespace std;
struct Node {
    int data;    Node* prev;    Node* next;
    Node(int val): data(val), prev(nullptr), next(nullptr) {}
};
```

```
struct DoublyLinkedList {
    Node* head;    Node* tail;
    DoublyLinkedList() : head(nullptr), tail(nullptr) {}
    void append(int val) {
        Node* newNode = new Node(val);
        if (head == nullptr) {
            head = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }

Void printList() {
    Node* current = head;
```