**M. Ali. Arslan**

**19F-0348**

**Assignment-01**

**Task # 01**

#include <iostream>

using namespace std;

class Node

{

public:

Node\* link;

int key;

int data;

Node(int k, int d)

: key(k),

data(d),

link(NULL)

{};

};

class LinkedList

{

Node\* head;

int nodeCount;

public:

LinkedList()

{

head = NULL;

nodeCount = 0;

}

Node\* nodeExists(int k)

{

Node\* temp = head;

if (temp != NULL) {

while (temp != NULL)

{

if (temp->key == k)

{

break;

}

temp = temp->link;

}

}

return temp;

}

void appendNode(Node\* n) {

if (nodeExists(n->key) != NULL) {

cout << "Node Already exists with key value : " << n->key << ". Append another node with different Key value" << endl;

}

else {

if (head == NULL) {

head = n;

nodeCount++;

}

else {

Node\* ptr = head;

while (ptr->link != NULL) {

ptr = ptr->link;

}

ptr->link = n;

nodeCount++;

}

}

}

void prependNode(Node\* n)

{

if (nodeExists(n->key) != NULL)

{

cout << "Cannot prepend node withs same key!" << endl;

}

else

{

if (head == NULL)

{

head = n;

nodeCount++;

}

else

{

n->link = head;

head = n;

nodeCount++;

}

}

}

void deleteNode(int k)

{

if (nodeExists(k) == NULL)

{

cout << "Node not found!\n Delete operation failed!\n";

}

else

{

if (head->link == NULL)

{

head = NULL;

cout << "Node deleted!" << endl;

nodeCount--;

}

else {

Node\* temp = head;

Node\* prev = NULL;

while (temp->key != k)

{

prev = temp;

temp = temp->link;

}

prev->link = temp->link;

temp = NULL;

nodeCount--;

}

}

}

void display()

{

Node\* temp = head;

while (temp != NULL)

{

cout << "( " << temp->key << ", " << temp->data << " )-->";

temp = temp->link;

}

cout << endl << endl;

}

LinkedList\* mergeList(LinkedList\* l1, LinkedList\* l2)

{

Node\* temp1 = l1->head;

Node\* prev = NULL;

while (temp1 != NULL)

{

prev = temp1;

temp1 = temp1->link;

}

prev->link = l2->head;

return l1;

}

void sorting()

{

int temp = 0;

for (int i = 0; i < nodeCount \* nodeCount; i++)

{

Node\* currentPtr = head;

Node\* nextPtr = head->link;

while (currentPtr != NULL && nextPtr != NULL)

{

if (currentPtr->data > nextPtr->data)

{

temp = currentPtr->data;

currentPtr->data = nextPtr->data;

nextPtr->data = temp;

}

currentPtr = currentPtr->link;

nextPtr = currentPtr->link;

}

}

}

};

int main()

{

Node a(1, 20);

Node b(2, 30);

Node c(3, 10);

Node d(4, 40);

Node e(5, 15);

Node f(6, 5);

Node g(7, 25);

LinkedList s1;

LinkedList s2;

s1.appendNode(&a);

s1.appendNode(&b);

s1.appendNode(&c);

s1.appendNode(&d);

s2.appendNode(&e);

s2.appendNode(&f);

s2.appendNode(&g);

s1.sorting();

s2.sorting();

LinkedList\* merged;

cout << "First Singly Linked List: \n";

s1.display();

cout << "Second Singly Linked List: \n";

s2.display();

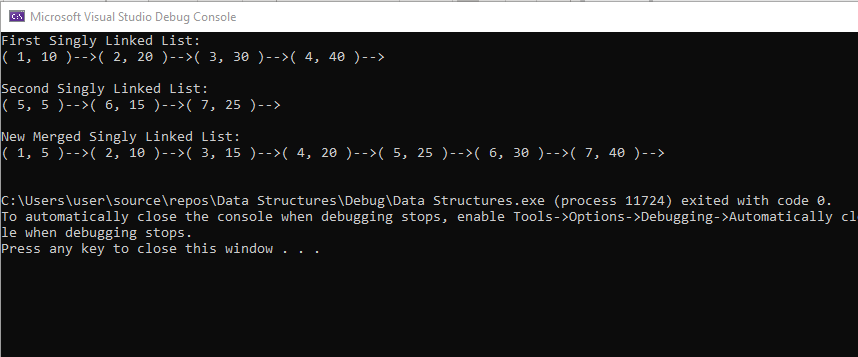
merged = s2.mergeList(&s1, &s2);

merged->sorting();

cout << "New Merged Singly Linked List: \n";

merged->display();

}



**Task # 02**

#include <iostream>

#include <string>

using namespace std;

class Node

{

public:

string patientName;

string CNIC;

string Disease;

string Date;

int roomNum;

Node\* next;

public:

Node()

{

patientName = " ";

CNIC = " ";

Disease = " ";

Date = " ";

roomNum = 0;

}

Node(string pName, string pCnic, string pDisease, string D, int rNum)

{

patientName = pName;

CNIC = pCnic;

Disease = pDisease;

Date = D;

roomNum = rNum;

}

};

class LinkedList

{

private:

Node\* head;

int patientCount;

int rooms[20];

Node\* nodeExists(int k)

{

Node\* temp = head;

if (temp != NULL) {

while (temp != NULL)

{

if (temp->roomNum == k)

{

break;

}

temp = temp->next;

}

}

return temp;

};

void appendNode(Node\* n) {

if (nodeExists(n->roomNum) != NULL) {

cout << "Room Number already alloted: " << n->roomNum << endl;

}

else {

if (head == NULL) {

head = n;

patientCount++;

}

else {

Node\* ptr = head;

while (ptr->next != NULL) {

ptr = ptr->next;

}

ptr->next = n;

patientCount++;

}

}

};

public:

LinkedList()

{

head = NULL;

patientCount = 0;

for (int i = 0; i < 20; i++)

{

rooms[i] = i + 1;

}

}

bool dischargePatient(string cnic)

{

if (head == NULL)

{

cout << "No patient to discharge!\n";

return false;

}

else

{

if (head->next == NULL)

{

cout << "\nPatient Name: " << head->patientName << endl <<

"Patient CNIC: " << head->CNIC << endl <<

"Patient Disease: " << head->Disease << endl <<

"Patient Room Number: " << head->roomNum << endl << endl;

cout << "Patient Discharged SuccessFully!" << endl;

rooms[head->roomNum - 1] = head->roomNum;

head = NULL;

patientCount--;

return true;

}

else {

Node\* temp = head;

Node\* prev = NULL;

while (temp->next != NULL)

{

if (temp->CNIC == cnic)

break;

prev = temp;

temp = temp->next;

}

cout << "\nPatient Name: " << temp->patientName << endl <<

"Patient CNIC: " << temp->CNIC << endl <<

"Patient Disease: " << temp->Disease << endl <<

"Patient Room Number: " << temp->roomNum << endl << endl;

cout << "Patient Discharged SuccessFully!" << endl;

rooms[temp->roomNum - 1] = temp->roomNum;

prev->next = temp->next;

temp = NULL;

patientCount--;

return true;

}

}

}

bool admitPatient()

{

// 20 Rooms Only

if (patientCount == 20)

{

cout << "Sorry, No room availible for your patient!" << endl;

return false;

}

else {

cout << "\t\t\t\t\tNew Patient!\n\n";

Node\* patient = new Node;

cout << "\nEnter patient name: ";

cin >> patient->patientName;

cout << "Enter patient CNIC: ";

cin >> patient->CNIC;

cout << "Enter Patient Disease: ";

cin >> patient->Disease;

cout << "Enter admit Date: ";

cin >> patient->Date;

for (int i = 0; i < 20; i++)

{

if (rooms[i] != 0)

{

patient->roomNum = rooms[i];

rooms[i] = 0;

break;

}

};

appendNode(patient);

cout << "\nPatient admitted successfully in room number: " << patient->roomNum << endl;

return true;

}

}

void searchPatient()

{

int selection = 0;

string patientCNIC, patientDisease, patientName;

int patientRoom;

Node\* temp = head;

bool found = false;

cout << "\nSearch patient by:\n";

cout << "1) CNIC\n2) Room Number\n3) Disease\n4) Name\n\n";

cin >> selection;

switch (selection)

{

case 1:

cout << "Input CNIC of patient!" << endl;

cin >> patientCNIC;

while (temp != NULL)

{

if (temp->CNIC == patientCNIC)

{

found = true;

cout << "Patient Name: " << temp->patientName << endl <<

"Patient CNIC: " << temp->CNIC << endl <<

"Patient Disease: " << temp->Disease << endl <<

"Patient Room Number: " << temp->roomNum << endl <<

"Patient Admit Date: " << temp->Date << endl << endl;

break;

}

temp = temp->next;

}

break;

case 2:

cout << "Input Room Number of patient!" << endl;

cin >> patientRoom;

while (temp != NULL)

{

if (temp->roomNum == patientRoom)

{

found = true;

cout << "Patient Name: " << temp->patientName << endl <<

"Patient CNIC: " << temp->CNIC << endl <<

"Patient Disease: " << temp->Disease << endl <<

"Patient Room Number: " << temp->roomNum << endl <<

"Patient Admit Date: " << temp->Date << endl << endl;

break;

}

temp = temp->next;

}

break;

case 3:

cout << "Input Disease of patient!" << endl;

cin >> patientDisease;

while (temp != NULL)

{

if (temp->Disease == patientDisease)

{

found = true;

cout << "Patient Name: " << temp->patientName << endl <<

"Patient CNIC: " << temp->CNIC << endl <<

"Patient Disease: " << temp->Disease << endl <<

"Patient Room Number: " << temp->roomNum << endl <<

"Patient Admit Date: " << temp->Date << endl << endl;

}

temp = temp->next;

}

break;

case 4:

cout << "Input Name of patient!" << endl;

cin >> patientName;

while (temp != NULL)

{

if (temp->patientName == patientName)

{

found = true;

cout << "Patient Name: " << temp->patientName << endl <<

"Patient CNIC: " << temp->CNIC << endl <<

"Patient Disease: " << temp->Disease << endl <<

"Patient Room Number: " << temp->roomNum << endl <<

"Patient Admit Date: " << temp->Date << endl << endl;

break;

}

temp = temp->next;

}

default:

break;

}

if (found == false)

{

cout << "Patient not found!" << endl;

};

}

};

int main()

{

LinkedList l1;

string cnic;

int selection = 0;

cout << "......................HOSPITAL MANAGEMENT SYSTEM......................" << endl;

while (selection != 4)

{

cout << "1) Admit Patient\n2) Discharge Patient\n3) Search Patient\n4) Exit\n\n Option:";

cin >> selection;

switch (selection)

{

case 1:

l1.admitPatient();

break;

case 2:

cout << "Enter CNIC of patient: ";

cin >> cnic;

l1.dischargePatient(cnic);

case 3:

l1.searchPatient();

break;

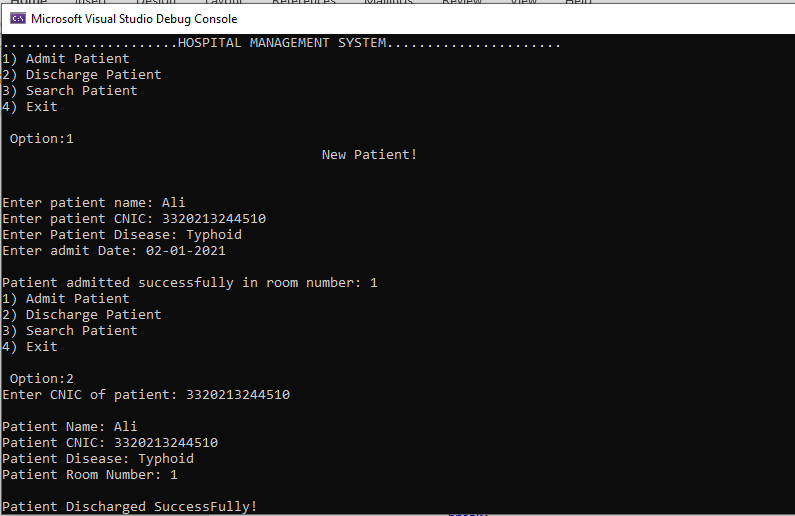
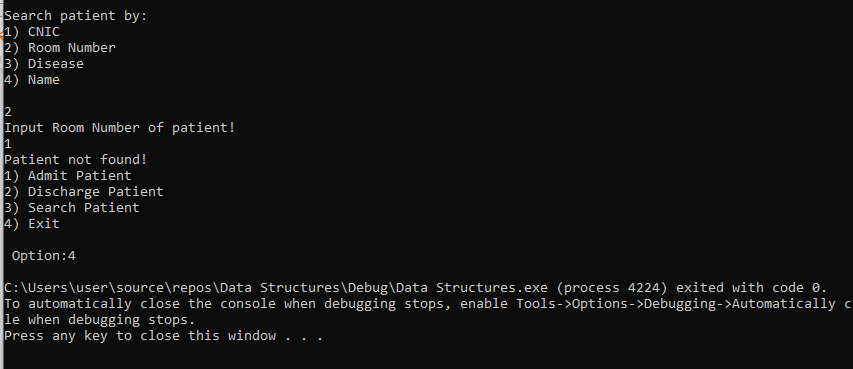
default:

break;

}

}

}

**Task # 04**

#include <iostream>

using namespace std;

class Node

{

public:

Node\* next;

int key;

int etime;

Node()

{

key = 0;

etime = 0;

next = NULL;

}

};

class LinkedList

{

public:

Node\* head;

int keys[50];

int c;

LinkedList()

{

for (int i = 0; i < 50; i++)

{

keys[i] = 0;

};

c = 0;

head = NULL;

};

bool nodeExists(int k)

{

bool res = true;

for (int i = 0; i < 50; i++)

{

if (k == keys[i])

{

res = false;

break;

}

}

return res;

};

void appendNode(Node\* n) {

if (nodeExists(n->key) == false) {

cout << "Node Already exists with key value : " << n->key << ". Append another node with different Key value" << endl;

}

else {

keys[c] = n->key;

if (head == NULL) {

head = n;

n->next = head;

}

else {

Node\* ptr = head;

while (ptr->next != head) {

ptr = ptr->next;

}

ptr->next = n;

n->next = head;

}

c++;

}

}

void printNodes()

{

Node\* temp = head;

for (int i = 1; i <= c; i++)

{

cout << "Processes in linked list: ";

for (int i = temp->key; i <= c; i++)

{

cout << i << " ";

}

cout << endl << endl;

cout << "Process # " << temp->key << " Execution Time: " << temp->etime << " seconds" << endl << endl;

cout << "Process # " << temp->key << " has been executed. Time Taken: " << temp->etime << endl << endl;

temp = temp->next;

}

}

};

int main()

{

LinkedList L;

int s = 0;

int count = 0;

cout << "Enter number of processes: ";

cin >> s;

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

{

Node\* n = new Node();

count++;

cout << "Enter Process # " << count << " execution time: ";

cin >> n->etime;

n->key = count;

L.appendNode(n);

}

L.printNodes();

}

