**Data Structures and Algorithms**

**Lab Journal - Lab 3**

Name: AREEBA HAKIM and AHMER ZUBERI

Enrollment #: 01-134182-012 and 01-134182-008

Class/Section: BSCS-3A

**Objective**

This lab session is aimed at enhancing the algorithmic development skills of the students by writing small utility functions for linked lists. In addition, the students will also implement the ‘Stack’ class using a linked list.

**Exercises**

Implement the following exercises.

**Exercise 1**

|  |
| --- |
| A stack can be implemented using a linked list. The first node can serve as the ‘top’ of Stack and ‘push’ and ‘pop’ operations can be implemented by adding and removing nodes at the head of the linked list. Implement the Stack class using a linked list and provide all the standard member functions. |

#include<iostream>

using namespace std;

struct node {

node\* next;

int data;

};

class stack {

public:

node\* head;

node\* temp;

node\* cur;

stack() { node\* head = NULL; node\* cur = NULL; node\* temp = NULL; }

void push(int addData) {

node\* add = new node;

add->next = NULL;

add->data = addData;

if (head != NULL) {

cur = head;

while (cur->next != NULL) {

cur = cur->next;

}

cur->next = add;

}

else

head = add;

}

void popp() {

while (cur->next != NULL)

cur = cur->next;

if (cur != NULL && head != NULL) {

temp = cur;

cout << endl;

cout << cur->data;

cout << " popped";

temp->data = NULL;

delete temp;

}

else

cout << "\n Empty";

}

void print() {

cur = head;

do {

cout << this->cur->data;

cout << endl;

cur = cur->next;

} while (cur != NULL);

}

};

void main() {

int ch, v;

stack obj;

cout << "\n Enter 1 to add value to push ";

cout << "\n Enter 2 to pop";

cout << "\n Enter 3 to display";

do {

cout << "\n Enter choice : ";

cin >> ch;

switch (ch) {

case 1:

{

cout << "\n Enter value : "; cin >> v;

obj.push(v);

break;

}

case 2:

{

obj.popp();

break;

}

case 3:

{

obj.print();

break;

}

default:

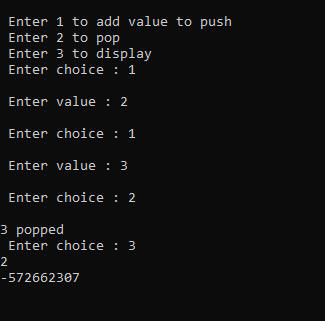
break;

}

} while (ch <= 3 && ch >= 1);

system("pause");

}



**Exercise 2**

|  |
| --- |
| Create an organized linked list in ascending order i.e., all the entries should be added in the list in ascending order. In order to do so the insert(int value) member function must be modified in the linked list class. An illustration of the working of the program is presented in the following.  List list ;  list.insert (5) ; //5  list.insert (2) ; //2 5  list.insert (7) ; //2 5 7  list.insert (3) ; //2 3 5 7 |

#include<iostream>

using namespace std;

class list {

public:

struct node {

node\* next;

int data;

};

node\* head;

node\* temp;

node\* cur;

list() { node\* head = NULL; node\* cur = NULL; node\* temp = NULL; }

void addnode(int addData, int c) {

node\* add = new node;

node\* tmp = head;

add->next = NULL;

add->data = addData;

if (head != NULL) {

cur = head;

if (c == 2 && cur->data > addData) {

add->next = head;

head = add;

}

else

if (cur->next == NULL&&cur->data < addData)

cur->next = add;

else

{

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

{

if (cur->data < addData && cur->next->data>addData) {

temp = cur;

cur = cur->next;

temp->next = add;

add->next = cur;

break;

}

else

cur = cur->next;

}

cur = cur->next;

}

}

else

head = add;

}

void print() {

cur = head;

do {

cout << this->cur->data;

cout << endl;

cur = cur->next;

} while (cur != NULL);

}

};

void main() {

char ch = 'y'; int check = 0;

int n;

list\* obj = new list;

while (ch == 'y') {

cout << "\n Enter value : ";

cin >> n;

check++;

obj->addnode(n, check);

cout << "\n Want to enter value y/n : ";

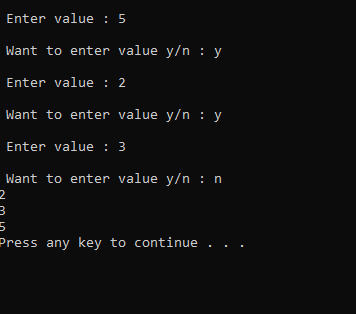
cin >> ch;

}

obj->print();

system("pause");

}



**Exercise 3**

|  |
| --- |
| Write C++ functions to perform following:   1. Concatenate two lists of characters together. 2. Find intersection of two lists. 3. Compare two strings (represented as linked lists) . The function should take two linked lists containing characters of two strings and compare each corresponding character in a lexicogrphical order. The function should return:  * 0 if both strings are same * 1 if first list is lexicographically greater * -1 if second string is lexicographically greater |

#include<iostream>

#include<string.h>

using namespace std;

class list {

public:

struct node {

node\* next;

string data;

};

node\* head;

node\* temp;

node\* cur;

list() { node\* head = NULL; node\* cur = NULL; node\* temp = NULL; }

void addnode(string addData) {

node\* add = new node;

add->next = NULL;

add->data = addData;

if (head != NULL) {

cur = head;

while (cur->next != NULL) {

cur = cur->next;

}

cur->next = add;

}

else

head = add;

}

void print() {

cur = head;

do {

if (cur == NULL)

break;

else

{

cout << cur->data;

cout << endl;

cur = cur->next;

}

} while (cur != NULL);

}

void intersect(list\* b) {

int check;

cur = head;

b->cur = head;

string s1, s2;

while (cur != NULL && b->cur != NULL) {

s1 = this->cur->data;

s2 = b->cur->data;

check = s1.compare(s2);

if (check == 0) {

cout << "\n" << this->cur->data << "\t";

cout << check;

}

else{

cout << "\n" << this->cur->data << "\t";

cout << check;

}

this->cur = this->cur->next;

}

while (b->cur != NULL && cur != NULL) {

check = cur->data.compare(b->cur->data);

if (check == 0) {

cout << "\n" << cur->data;

}

b->cur = b->cur->next;

}

}

};

void main() {

int ch, n;

string v;

list\* obj = new list;

list\* obj1 = new list;

cout << "\n Enter 1 to add string ";

cout << "\n Enter 2 to intersect lists ";

cout << "\n Enter 3 to display";

do {

cout << "\n Enter choice : ";

cin >> ch;

switch (ch) {

case 1:

{

cout << "\n Enter value : "; cin >> v;

cout << "\n In List 1or 2(1/2) : "; cin >> n;

if (n == 1)

{

obj->addnode(v);

}

if(n==2) {

obj1->addnode(v);

}

break;

}

case 2:

{

obj->intersect(obj1);

break;

}

case 3:

{

obj->print();

obj1->print();

break;

}

default:

break;

}

} while (ch <= 3 && ch >= 1);

system("pause");

}

**Implement the given exercises and get them checked by your instructor. If you are unable to complete the tasks in the lab session, deposit this journal alongwith your programs (printed or handwritten) before the start of the next lab session.**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Exercise** | **Checked By:** |
| 1. | Exercise 1 |  |
| 2. | Exercise 2 |  |
| 3. | Exercise 3a. |  |
| 4. | Exercise 3b. |  |
| 5. | Exercise 3c. |  |

+++++++++++++++++++++++++