CS 124 Test 3 Spring 2008 Thursday, April 3

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ CWID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Section: (circle one) 11:00 12:30 2:00

1. What is missing? [ at (1), (2), and (3) ]. What is output when the code is fixed? (15 points)

#include <iostream>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_// (1)

using namespace std;

int main ( ) {

Node \*a, \*b, \*c;

a = new Node(5);

b = new Node(10, a);

c = new Node(15, b);

Node \*temp;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // (2)

while (temp != 0) {

int x = temp->getData( );

temp->setData(3\*x);

cout << temp->getData() << endl;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // (3)

}

return 0;

}

2. Using class Node below, write a program that creates a “list” containing the data Chicago, Detroit, Nashville, Boston in this order. Use dynamic memory allocation. Print the “list”. (15 points)

class Node {

string data;

Node \*next;

public:

Node( ) : data(0), next(0) { }

Node(string e) : data(e), next(0) { }

Node(string e, Node \*n) : data(e), next(n) { }

string getData(void) { return data; }

Node \*getNext(void) { return next; }

void setData(string e){ data = e; }

void setNext(Node\* n){ next = n; }

};

3. For the Node class above and for the List class below, complete the definition for class method pop\_pop\_back ( ). [ at (1), (2), and (3) ] (15 points)

class List {

Node \*start;

public:

List();

void push\_front(string e);

void push\_back(string e);

void pop\_pop\_back( ); // removes the last two list elements

int size(void) const;

Node \*find(string);

void insert(Node \*, string); // inserts **BEFORE** the location

void print(void);

};

void List::pop\_pop\_back( ){

Node \*t1=start, \*t2=start, \*t3;

if(size()>2){

while(t1->getNext()!=0){

// (1) what goes in here? Multiple lines.

}

delete t1; delete t2; t3->setNext(0);

}

else if(size()==2){

// (2) what line of code is needed here?

t1=t1->getNext(); delete t1; delete t2;}

else if( ?????? ){ // (3) what goes in place of ??????

start=0;

delete t1;}

}

4. For the list class from # 3 above write the definition (outside of the declaration) of insert. (20 points)

5. Assuming a standard Node class and needed headers, what is output by the following? (15 points)

class List {

Node \*start;

public:

List();

List(const List& l);

void push\_back(int e);

void print(void) const;

};

List::List() : start(0) { }

List::List(const List& l): start(0){

cout << "copy" << endl;

Node \*temp = l.start;

while (temp){

push\_back(temp->getData());

temp = temp->getNext();}

}

void List::push\_back(int e) {

Node \*newone = new Node(e);

if (start == 0)

start = newone;

else {

Node \*temp = start;

while (temp->getNext() != 0)

temp = temp->getNext();

temp->setNext(newone);

}

return;

}

void List::print(void) const {

Node \*temp = start;

while (temp != 0) {

cout << temp->getData() << " ";

temp = temp->getNext();}

return;

}

int function1(List& k){cout << endl << "in fun1" << endl; return 0;}

int function2(List k){cout << endl << "in fun2" << endl; return 0;}

int main() {

List a;

a.push\_back(23); a.push\_back(35);a.push\_back(27);

List b(a); b.print();

function1(a);

function2(b);

return 0;}

6. What is the output of the following program? (20 points)

#include <iostream>

#include <string>

using namespace std;

class Node {

int data;

Node \*next;

public:

Node() : data(0), next(0) { }

Node(int e) : data(e), next(0) { }

Node(int e, Node\* n) : data(e), next(n) { }

friend class List;

friend ostream& operator<<(ostream&, const List&);

friend ostream& operator<<(ostream&, const Node&);

};

ostream& operator<<(ostream& os, const Node& n){

os << n.data; return os;}

class List {

Node \*start;

public:

List() : start(0) { }

~List();

void push\_back(int);

List& operator=(const List&);

friend ostream& operator<<(ostream&, const List&);

};

List::~List( ) {

if (start != 0) {

Node \*t1, \*t2;

t1 = start;

while (t1 != 0) {

t2 = t1; t1 = t1->next;

cout << "deleting " << t2->data << endl;

delete t2; }}

}

void List::push\_back(int e) {

Node \*newone = new Node(e);

if (start == 0)

start = newone;

else {

Node \*temp = start;

while (temp->next != 0) temp = temp->next;

temp->next=newone;

}

return;

}

ostream& operator<<(ostream& os, const List& l) {

Node \*temp = l.start;

while (temp) { os << \*temp << " "; temp = temp->next; }

return os;}

List& List::operator=(const List& a) {

if (this == &a) return \*this;

else if (start != 0) {

Node \*t1, \*t2;

t1 = start;

while (t1 != 0) { t2 = t1; t1 = t1->next; delete t2; }

}

start = 0;

Node \*temp = a.start; cout << "assign" << endl;

while (temp) { push\_back( temp->data ); temp = temp->next; }

return \*this;

}

int main( ) {

List a, b;

a.push\_back(1); a.push\_back(5);

b = a; b.push\_back(7);

cout << a << b << endl;

return 0;

}