C++ Homework

CS032 2008

Due 4/10 at 11:59

All homeworks are due on 4/10 at 11:59 in the CS 32 bin on the second floor. No late homeworks are accepted.
Please include your login name on each piece of paper you hand in, and please staple your pages together before handing in.

Before attempting this homework you are highly encouraged to understand the material the basics of C++. This mini course, blatantly stolen from cs 123, might be useful:

http://www.cs.brown.edu/courses/csci1230/resources/c++_mini_course.pdf

1 Namespaces [10 pts]

Suppose that you have deemed it impossible to change the names of either of these classes (Sometimes libraries will have classes with duplicate names, but hopefully they will use good practices just like you. "Hopefully" being the key word). Use namespaces to avoid conflict and fix the problem below:

```
//header1.h
#ifndef HEADER1
#define HEADER1

class List {
        public:
                void add(int item);
                void remove_front();

};

#endif
```

```
//header2.h
#ifndef HEADER2
#define HEADER2

class List {
        public:
                void insert(int item);
                void delete_front();
};

#endif
```

Continued on next page.

```
#include "header1.h"
#include "header2.h"

int main()
{
        //stuff
        return 0;
}
```

2 New and Delete [20 pts]

a. [15 pts] Write a remove function for the following linked list implementation.

```
//list.h
#ifndef LIST
#define LIST

struct ListItem {
        ListItem *next;
        int myInt;
};

class List {
        public:
                List();
                void add(int item);
                void printlist();
                void remove_front();
        private:
                ListItem *front;
};

#endif
```

Continued on next page.

```
//list.cpp
#include "List.h"
#include <iostream>
using namespace std;
List::List()
{
        front = NULL;
}
void List::add(int item) {
        ListItem *element = new ListItem();
        element->myInt = item;
        element->next = front;
        front = element;
}
void List::printlist() {
        ListItem *ph = front;
        while(ph != NULL) {
                cout << ph->myInt << endl;
                ph = ph->next;
        }
}
void List::remove_front() {
        //This should remove the first item in the list.
        //If the list is empty do nothing.
}
```

b. [5 pts] Why, inspite of my perfect remove_front, does valgrind still tell me that I leak memory when I run the following code? What would I need to do to fix it?

```
#include "list.h"

int main()
{
        List list;
        list.remove_front();
        list.remove_front();
        list.printlist();

        list.add(1);
        list.add(3);
        list.add(5);
        list.printlist();

        list.remove_front();
        list.remove_front();
        list.printlist();

        return 0;
}
```

3 Header Files vs. Cpp Files [10 pts]

When should you put function bodies in C++ header files?

4 Overloading in C++ vs. Java [25 pts]

a. [5 pts] Explain the difference between virtual and nonvirtual functions. Is there a java equivalent to nonvirtual functions? If so, what is it?

b. [5 pts] Explain the difference between virtual and pure virtual functions. Is there a java equivalent to pure virtual functions? If so, what is it?

c. [5 pts] Explain why inline functions cannot be virtual. Is there a java equivalent to inline functions? If so, what is it?

d. [10 pts] Given the following listing, what does an instantiation of B do, and how is it different from what a similar piece of code in Java would do?

```
//A.h
#ifndef A_H
#define A_H

class A {
        public:
           A();
                virtual void printme();
};
#endif
```

```
//A.cpp
#include "A.h"
#include <iostream>
using namespace std;

A::A()
{
   printme();
}

void A::printme()
{
   cout << "A" << endl;
}
```

Continued on next page.

```
//B.h
#ifndef B_H
#define B_H

#include "A.h"

class B : public A {
        public:
           B();
                virtual void printme();
};
#endif
```

```
//B.cpp
#include "B.h"
#include <iostream>
using namespace std;

B::B()
{
}

void B::printme()
{
   cout << "B" << endl;
}
```

5 Inheritance and Polymorphism [20 pts]

Implement a simple geometry library using inheritance that allows one to perform calculations on right triangles, rectangles, circles. The shapes should expose functionality as follows:

```
All shapes
        void setLocation(double x, double y);
        double getXLocation();
        double getYLocation();
        double getArea();
        double getPerimeter();
Right Triangles and Rectangles
        void setHeight(double h);
        void setBase(double b);
        double getHeight();
        double getBase();
Circles
        void setRadius(double r);
        double getRadius();
```

6 Operator Overriding [15 pts]

Override the << operator in the list from question 2 so that the following statement prints your whole list. You may need to make changes to the header file.

```
List myList = new List();
cout << myList << endl;
```

Make sure to be clear on where your code should be inserted (explanatory comments, recopying and updating the header, etc. as long as it is clear what you intend)