Stevie Parris

Written Problems 6

1. When you print out a pointer, with the method described, it prints the address of the different locations. All the pointer values are different because they all point to different addresses. The numbers change each time you run the program because you’re storing the values in different locations.

x = 6278664

q = 6278668

x + 1 = 6278728

q + 1 = 6278736

1. Exercise R7.8. Which of the following assignments are legal in C++?

void f(int p[]) {

int\* q;

const int\* r;

int s[10];

p = q; FAILS DUE TO UNINITIALIZE

p = r; FAILS

p = s; FAILS DUE TO UNINITIALIZE

q = p; SUCEED

q = r; FAILS

q = s; SUCEED

r = p; SUCEED

r = q; SUCEED

r = s; SUCEED

s = p; FAILS

s = q; FAILS

s = r; FAILS

}



***(8 points)*** Let **v** be a **vector<Thingie\*>**, so that each element **v[i]** contains a pointer to a **Thingie**. If **p** is a **vector<Thingie\*>::iterator**, answer the following questions:

* 1. what type is **p**?

p is a Thingie object

* 1. what type is **\*p**?

It’s a pointer Thingie type

* 1. what code provides the address of the actual **Thingie**?

p

* 1. what code provides the actual **Thingie**?

\*p



***(8 points)*** **Chess**. Suppose you are programming a system that plays chess. You decide that chess pieces can be designed as classes, using inheritance. Notice that there are some characteristics that all pieces have in common (for instance, each of them has a name), and some characteristics differ for each piece (although each can move, the types of moves a piece can make differ). Show a design for chess pieces by giving the attributes and methods of a base class and showing the derived classes. Will any of these be virtual methods? *You don't have to write any code for this problem, but should specify the base and derived classes, properties and methods.*

The base class would be “Chess Piece”, and it would have derived classes such as pawn, knight, queen, etc. Each of these have the method move and captured and attributes such as location, color, and more. Since each piece will have a movement method, it will need to be declared virtually to differentiate between the base class and the inherited class.

1. This is because both iostream and ifstream are derived from istream class.

Exercise R8.6. Suppose the class D inherits from B.

Which of the following assignments are legal?

B b;

D d;

B\* pb;

D\* pd;

a. b = d; VALID

b. d = b; INVALID

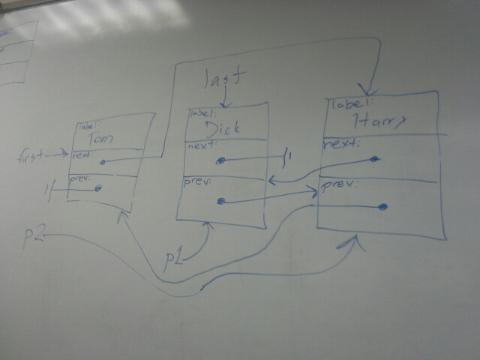
c. pd = pb; INVALID

d. pb = pd; VALID

e. d = pd; INVALID cannot assign pointer to location

f. b = \*pd; VALID

g. \*pd = \*pb; INVALID



1. ***Deep vs. Shallow Assignment***:

If **myPtr** and **yourPtr** are pointers to linked lists of some type, explain what happens if the default assignment operator is used to assign **myPtr** to **yourPtr**. How the assignment operator for this type must be modified in order to make sure

(a) No memory is leaked, and

(b) A copy of the entire list is made?

Do any problems occur if **myPtr** and **yourPtr** both point to the same list initially? Explain how to handle this problem.

If you assign myPtr to yourPtr, myPtr points to the same location as yourPtr. The object myPtr was assigned to is now not pointed to, and therefore we don’t know where it is in the memory, so memory is leaked. In order to prevent leaking of memory, you must call the destructor for the pointer, which will delete each successive object, using the link, and deleting each object. If you delete the objects in the list for myPtr, now yourPtr is a dangling pointer, and so you could eliminate this problem by deleting one pointer and then treating all the items in the list as a single pointer.