1. (40 points) Is there any output for the following program? If there is output, what is it? Class string is not well written. What are the problems with this class? Fix the problems.

```
#include <cstring>
#include <iostream>
class string {
public:
  string() : buf(new char[1]) { buf[0] = '\0'; }
  string(const char* s) : buf(new char[strlen(s)+1]) { strcpy(buf, s); }
  ~string() { delete [] buf; }
  string& operator=(const string& rhs) {
    if ( this == &rhs ) return *this;
    delete [] buf;
    buf = new char[strlen(rhs.buf)+1];
    strcpy(buf, rhs.buf);
    return *this;
  }
  const char* getBuf() const { return buf; }
  void setBuf(const char* b) {
    if ( b == buf || b == NULL ) return;
    delete [] buf;
    buf = new char[strlen(b)+1];
    strcpy( buf,  b );
  }
private:
  char * buf;
  string(const string&);
};

int main() {
  string x("cat"), y;
  y = x;
  //x.setBuf("What do you wish my Thane");
  x.setBuf(x.getBuf());
  std::cout << x.getBuf() << std::endl;
}
```

2. (10 points) Write an assignment operator for class Student.

```
#include <cstring>
class Student : public Person {
public:
  Student() : Person(), gpa(0) {}
  Student(const char* n, float g) : Person(n), gpa(g) { }
  Student(const Student& s);
  Student& operator=(const Student& rhs);
private:
  float gpa;
};
```

3. (20 points) Give the output for the following program.

```
#include <iostream>
#include <vector>
const int MAX = 3;
class Number {
public:
  Number() : number(0) { std::cout << "default" << std::endl; }
  explicit Number(int n) : number(n) {
    std::cout << "convert: " << n << std::endl;
  }
  Number(const Number& a) : number(a.number) {
    std::cout << "copy: " << a.number << std::endl;
  }
  Number& operator=(const Number& rhs) {
    if ( this != &rhs ) { number = rhs.number; }
    std::cout << "assign" << std::endl;
    return *this;
  }
  int getNumber() const { return number; }
private:
  int number;
};

void print(const std::vector<Number> & vec) {
  for (unsigned int i = 0; i < vec.size(); ++i) {
    std::cout << vec[i].getNumber()  << ", ";
  }
  std::cout << std::endl;
}

void init(std::vector<Number> & vec) {
  for (unsigned int i = 0; i < MAX; ++i) {
    vec.push_back( Number(i+1) );
  }
}

int main() {
  std::vector<Number> vec;
  vec.reserve(3);
  init(vec);
  vec.push_back( Number(4) );
  std::cout << "SIZE: " << vec.size() <<  std::endl;
  std::cout << "CAP:  " << vec.capacity() <<  std::endl;
  print(vec);
}
```

4. (10 points) The header file for class Manager is listed below, with declarations for an SDL_Surface on line #20, a declaration of a Frame on line #21, and functions draw and update on lines 29 and 30.

   (a) Why are the declarations of the SDL_Surface and Frame found in the Manager class rather than the Sprite class.

```
1  #include <SDL.h>
2  #include "ioManager.h"
3  #include "clock.h"
4  #include "sprite.h"
5
6  class Manager {
7  public:
8    Manager ();
9    ~Manager ();
10   void play();
11 private:
12   const bool env;
13   const IOManager* io;
14   Clock& clock;
15   SDL_Surface * const screen;
16   int backRed;
17   int backGreen;
18   int backBlue;
19
20   SDL_Surface* orbSurface;
21   const Frame * const orbFrame;
22   Sprite orb;
23
24   bool makeVideo;
25   int frameCount;
26   std::string username;
27   int frameMax;
28   const std::string TITLE;
29   void draw() const;
30   void update();
31   Manager(const Manager&);
32   Manager& operator=(const Manager&);
33   void drawBackground() const;
34 };
```

5. (20 points) The following questions refer to the Items in the Meyer's book.

   (a) For class Manager, shown on the previous page, why are the functions on lines 31–33 declared in private?

   (b) Why should you declare destructors to be virtual in polymorphic base classes?

   (c) Why should you make an assignment operator return a reference to *this?

   (d) Why should you never call a virtual function from a base class constructor? (Please answer in terms of the order that base classes and derived classes are constructed).