1. (15 points) The following program prints “cat” and then crashes with a *double free* error. Write a function that will eliminate this error.

```
#include <cstring>
#include <iostream>
using std::cout;  using std::endl;
class string {
public:
   string(const char* s) : buf(new char[strlen(s)+1]) { strcpy(buf, s);}
   ~string() { delete [] buf; }
   const char* getBuf() const { return buf; }
private:
   char * buf;
};

int main() {
  string a("cat"), b = a;
  cout << a.getBuf() << endl;
}
```

2. (15 points) The following program prints “cat” and then crashes with a *double free* error. Write a function that will eliminate this error.

```
#include <cstring>
#include <iostream>
using std::cout;  using std::endl;
class string {
public:
   string() : buf(new char[1]) { strcpy(buf, "dog"); }
   string(const char* s) : buf(new char[strlen(s)+1]) { strcpy(buf, s);}
   ~string() { delete [] buf; }
   const char* getBuf() const { return buf; }
private:
   char * buf;
};

int main() {
  string a("cat"), b;
  b = a;
  cout << b.getBuf() << endl;
}
```

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

```cpp
#include <iostream>
class Binary {
public:
   Binary() : number(0), myCount(count) {
     ++count;
     std::cout << "default: " << myCount << std::endl;
   }
   Binary(int n) : number(n), myCount(count) {
     ++count;
     std::cout << "convert: " << myCount << std::endl;
   }
   Binary(const Binary& bin) : number(bin.number), myCount(count) {
     ++count;
     std::cout << "copy: " << myCount << std::endl;
   }
   ~Binary() { std::cout << "destructor: " << myCount << std::endl; }
   Binary& operator=(const Binary&) {
     std::cout << "assignment" << std::endl;
     return *this;
   }
   int getNumber() const { return number; }
   void increment() { ++number; }
private:
   int number;
   int myCount;
   static int count;
};

Binary increment(Binary bin) {
  bin.increment();
  return bin;
}
int Binary::count = 0;
int main() {
  Binary a(17), b = a;
  b = increment(a);
}
```

4. The following program compiles and executes.

   (a) Give the output for the program. (5 points)

   (b) What public inlined functions does $C^{++}$ silently write? (10 points)

```
#include <iostream>
class Binary {
public:
   int getNumber() const { return number; }
   void setNumber(int n) { number = n; }
private:
   int number;
};

int main() {
  Binary a, b = a;
  a.setNumber(19);
  std::cout << b.getNumber() << std::endl;
}
```

5. (5 points) The following program does **not** compile, and the compiler issues an error message indicating that the compiler did not write a default constructor. Why didn't the compiler write a default constructor?

```
 1 #include <iostream>
 2 class Binary {
 3 public:
 4   Binary(int n) : number(n) {}
 5   int getNumber() const { return number; }
 6   void setNumber(int n) { number = n; }
 7 private:
 8   int number;
 9 };
10
11 int main() {
12   Binary a, b(19);
13   std::cout << b.getNumber() << std::endl;
14 }
main.cpp:12:10: error: no matching constructor for initialization of 'Binary'
  Binary a, b(19);
         ^
```

6. (20 points) Make class Manager a Singleton.

```
#include <iostream>
#include <vector>
class Sprite {};

class Manager {
public:
  Manager() : sprites() { }
  void manage() {
    // By doing nothing
  }
private:
  std::vector<Sprite> sprites;
  Manager(const Manager&);
  Manager& operator=(const Manager&);
};

int main() {
  Manager manager;
  manager.manage();
}
```

7. (10 points) Give the output for the following program:

```
#include <cstdio>
#include <iostream>
using std::cout; using std::endl;

class Student {
public:
  Student() { cout << "default" << endl; }
  Student(const char *b) { cout << "convert" << endl; }
  Student& operator=(const Student&) {
    cout << "assign" << endl;
    return *this;
  }
private:
  char *buf;
};

class TestStudent {
  Student str;
public:
  TestStudent(const char* s) {
    str = s;
  }
};

int main() {
  TestStudent t1("cat");
}
```

8. (5 points extra) Consider the following implementation of an assignment operator for string. Does this implementation correctly handle the problem of assignment to self? Explain.

```
string& string::operator=(const string& rhs) {
  string temp(rhs);
  delete [] buf;
  buf = new char[strlen(temp.buf)+1];
  strcpy(buf, temp.buf);
  return *this;
}
```