1. Write all code necessary to convert Random into a GoF singleton. Make sure the program compiles and runs.

```
#include <cstdlib>  // for rand()
#include <iostream>

class Random {
public:
  Random() {
    int seed = time(0);
    srand(seed);
  }
  int operator()(int a, int b) {
    return (rand() % b) + a;
  }
private:
  Random(const Random&);
  Random& operator=(const Random&);
};
int main() {
  Random random;
  std::cout << random(1,100) << std::endl;
}
```

2. Give the output for the following program.

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

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

3. Give the output for the following program.

```
#include <cstring>
#include <iostream>
class Student {
public:
   Student(const char* n) : name(new char[strlen(n)+1]) { strcpy(name, n); }
   const char* getName() const { return name; }
   void setName(const char* n) {
     delete [] name;
     name = new char[strlen(n)+1];
     strcpy(name, n);
   }
private:
   char * name;
};

int main() {
  Student a("John"), b("Mary");
  b = a;
  b.setName("Sam");
  std::cout << a.getName() << std::endl;
}
```

4. (30 points) Write a copy constructor, assignment operator, and output operator for class Student in the previous problem.

5. Give the output for the following program.

```
#include <iostream>
#include <vector>
class A {
public:
  A() {}
  virtual void foo() const {
    std::cout << "I'm foo in A" << std::endl;
  }
  void bar() const {
    std::cout << "I'm bar in A" << std::endl;
  }
};
class B : public A {
public:
  B() : A() {}
  virtual void foo() const {
    std::cout << "I'm foo in B" << std::endl;
  }
  void bar() const {
    std::cout << "I'm bar in B" << std::endl;
  }
};
int main() {
  std::vector<A*> vec;
  vec.push_back( new B );
  vec[0]->foo();
  vec[0]->bar();
}
```

6. Give the output for the following program. Then add code so that there are no memory leaks.

```
#include <iostream>
class B {
public:
  B(int n) : number(n + '0') {}
  char getNumber() const { return number; }
private:
  char number;
};
class A {
public:
  A(int n) : b(new B(n)) {}
  char getB() const { return b->getNumber(); }
private:
  B* b;
};
int main( ){
  A* a = new A(7);
  std::cout << a->getB() << std::endl;
}
```

7. Give the output for the following program.

```
#include <iostream>
#include <cstdlib>
#include <vector>
const int MAX = 2;
class A{
public:
  A() { std::cout << "default" << std::endl; }
  A(const A&) { std::cout << "copy" << std::endl; }
};
template <typename T>
void print(std::vector<T>& vec) {
  std::cout << "size: " << vec.size() << '\t'
            << "cap: " << vec.capacity() << std::endl;
}
int main() {
  std::vector<int> vec1;
  std::vector<int> vec2;
  vec2.reserve(MAX);
  vec1.push_back(rand() % 100);
  vec2.push_back(rand() % 100);
  std::vector<A> vec3(MAX);
  vec3.push_back( A() );
  print(vec1);
  print(vec2);
  print(vec3);
}
```

8. The following program puts 100 letters into a vector. Write function eraseVowels so that it removes all *vowels* from the vector.

```
#include <iostream>
#include <vector>
#include <cstdlib>
const int MAX = 100;
const int LETTERS = 26;
void init(std::vector<char> & vec) {
  for (unsigned int i = 0; i < MAX; ++i) {
    vec.push_back( rand() % LETTERS + 'A' );
  }
}
void print(const std::vector<char> & vec) {
  for (unsigned int i = 0; i < vec.size(); ++i) {
    std::cout << vec[i]  << " ";
  }
  std::cout << std::endl;
}
int main() {
  std::vector<char> vec;
  init(vec);
  eraseVowels(vec);
  print(vec);
}
```