1. (20 points) Give the output for the following program:

```
#include <iostream>
#include <cstring>

class Sprite {
public:
  Sprite(const char* n) : name(new char[strlen(n)+1]) {
    strcpy(name, n);
  }
  const char* getName() const { return name; }
  const char& operator[](int index) const {
    std::cout << "const[] " << name[index] << std::endl;
    return name[index];
  }
  char& operator[](int index) {
    std::cout << "non-const[] " << name[index] << std::endl;
    return name[index];
  }
private:
  char* name;
};
void printLetter(const Sprite& sprite, int n) {
  const char letter = sprite[n];
}

int main() {
  Sprite sprite("redorb");
  printLetter(sprite, 0);
}
```

2. (40 points) Write an overloaded bracket and multiplication operator for class Rational.

```
#include <iostream>

class Rational {
public:
  Rational(int numerator = 0, int denominator = 1) :
    n(numerator), d(denominator)
  { }
private:
  int n, d;
};
std::ostream& operator<<(std::ostream& out, const Rational& r) {
  return out << r[0] << '/' << r[1];
}

int main() {
  Rational rat1(2,3), rat2(3, 4);
  std::cout << rat1 << std::endl;
  std::cout << rat1*rat2 << std::endl;
}
```

3. (40 points) Write a copy constructor and assignment operator for class ShootingSprite.

```
#include <iostream>
#include <cstring>

class Drawable {
public:
  Drawable(const char* n) : name(new char[strlen(n)+1]) {
    strcpy(name, n);
  }
  virtual void shoot() const = 0;
private:
  char* name;
};

class ShootingSprite : public Drawable {
public:
  ShootingSprite(const char* name, const char* b) :
    Drawable(name),
    bulletName(new char[strlen(b)+1]) {
    strcpy(bulletName, b);
  }
  virtual ~ShootingSprite() { delete [] bulletName; }
  virtual void shoot() const { std::cout << bulletName << std::endl; }
private:
  char* bulletName;
};

int main() {
  ShootingSprite d("redorb", "egg");
  ShootingSprite* e = new ShootingSprite("greenorb", "bullet");
  d.shoot();
  e->shoot();
  d = *e;
  d.shoot();
  delete e;
}
```