1. (10 points) Give the output for the following program.

```
#include <iostream>

void h(int z) {
  ++z;
}

void g(int& y) {
  ++y;
  h(y);
}

void f(int& x) {
  ++x;
  g(x);
  std::cout << x << std::endl;
}

int main() {
  int q = 7;
  f(q);
  std::cout << q << std::endl;
}
```

9

9

---

2. (10 points) Give the output for the following program.

```
#include <iostream>

void f(int n) {
  static int x = n;
  ++x;
  std::cout << x << std::endl;
}

int main() {
  f( 5%2?7:8 );
  f(17);
}
```

8

9

3. (10 points) What constructors are used on line 13? Write an overloaded output operator so that line 14 prints 9.

```
1  #include <iostream>
2  #include <cstring>
3
4  class A {
5  public:
6    A(int n) : number(n) {}
7    int getNumber() const { return number; }
8  private:
9    int number;
10 };
11
12 int main() {
13   A a(9), b = a;
14   std::cout << b << std::endl;
15 }
```

conversion, copy

```
std::ostream& operator<<(std::ostream& out, const A& a) {
  return out << a.getNumber();
}
```

---

4. (10 points) We discussed canonical form for classes, also known as the Rule of Three. What does canonical form mean? There are two classes listed in this exam: A and Game; which of these classes should be in canonical form?

Canonical form means that if a class contains pointer data then the programmer should supply (1) destructor, (2) copy constructor, and (3) copy assignment.

Game should be in canonical form.

5. (60 points) For class Game, partially listed below, write functions for default, conversion, copy, assignment, destructor, and getName(). Use initialization lists wherever applicable. Use const as much as possible.

```
1  #include <iostream>
2  #include <cstring>
3
4  class Game {
5  public:
6  private:
7    char* name;
8  };
9
10 int main() {
11   Game left4Dead, massEffect("Mass Effect"), ms = massEffect;
12   left4Dead = "Left4Dead";
13   std::cout << ms.getName() << std::endl;
14   std::cout << left4Dead.getName() << std::endl;
15 }
```

```
class Game {
public:
  Game() : name(new char[1]) { name[0] = '\0'; }
  Game(const char* n) : name(new char[strlen(n)+1]) {
    strcpy(name, n);
  }
  Game(const Game& vg) : name(new char[strlen(vg.name)+1]) {
    strcpy(name, vg.name);
  }
  ~Game() { delete [] name; }
  const char* getName() const { return name; }
  Game& operator=(const Game& rhs) {
    if ( this == &rhs ) return *this;
    delete [] name;
    name = new char[strlen(name)+1];
    strcpy(name, rhs.name);
    return *this;
  }
private:
  char* name;
};
```