1. (15 points) Give the output for the following program:

```
#include <iostream>
class Title {
public:
  Title()              { std::cout << "default" << std::endl;     }
  Title(const char*)   { std::cout << "convert" << std::endl;     }
  Title(const Title&)  { std::cout << "copy" << std::endl;        }
  ~Title()             { std::cout << "destructor" << std::endl; }
  Title& operator=(const Title&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
};

class NPC {
public:
  NPC(const char* t) {
    title = t;
  }
private:
  Title title;
};

int main() {
  NPC* npc = new NPC("Jarl of Whiterun");
}
```

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

```
#include <iostream>
class string {
public:
  string()              { std::cout << "default" << std::endl;     }
  string(const char*)   { std::cout << "convert" << std::endl;     }
  string(const string&) { std::cout << "copy" << std::endl;        }
  ~string()             { std::cout << "destructor" << std::endl; }
  string& operator=(const string&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
};
int main() {
  string* x = new string("cat");
  string y = *x;
}
```

3. (10 points) Give the output for the following program:

```
1  #include <iostream>
2  #include <vector>
3  const int MAX = 2;
4
5  class Number {
6  public:
7    Number() : number(0) { std::cout << "default" << std::endl; }
8    explicit Number(int n) : number(n) {
9      std::cout << "convert: " << n << std::endl;
10   }
11   Number(const Number& a) : number(a.number) {
12     std::cout << "copy: " << a.number << std::endl;
13   }
14   Number& operator=(const Number& rhs) {
15     number = rhs.number;
16     std::cout << "assign" << std::endl;
17     return *this;
18   }
19   int getNumber() const { return number; }
20 private:
21   int number;
22 };
23
24 void print(const std::vector<Number> & vec) {
25   for (unsigned int i = 0; i < vec.size(); ++i) {
26     std::cout << vec[i].getNumber()  << ", ";
27   }
28   std::cout << std::endl;
29 }
30
31 void init(std::vector<Number> & vec) {
32   for (unsigned int i = 0; i < MAX; ++i) {
33     vec.push_back( Number(i+1) );
34   }
35 }
36
37 int main() {
38   std::vector<Number> vec;
39   vec.reserve(2);
40   init(vec);
41   vec.push_back( Number(99) );
42   std::cout << "SIZE: " << vec.size() <<  std::endl;
43   std::cout << "CAP:  " << vec.capacity() <<  std::endl;
44   print(vec);
45 }
```

4. (5 points) Class Number contains a method, lines 6–8, that overloads the output operator. Write code on line #17 to use this output method to display Number variable number:

```
1  #include <iostream>
2  class Number {
3  public:
4    Number(int n) : number(n) {}
5    int getNumber() const { return number; }
6    std::ostream& operator<<(std::ostream& out) {
7      return out << number;
8    }
9  private:
10   int number;
11   Number& operator=(const Number&);
12   Number(const Number&);
13 };
14 int main( ) {
15   Number number(17);
16
17   // ----------------------------------------
18 }
```

5. (10 points) Give the output for the following program:

```
1  #include <iostream>
2  #include <cstring>
3  #include <string>
4  class A {
5  public:
6    A(const std::string& n) : name(n) {}
7    ~A() { std::cout << "base" << std::endl; }
8    virtual void f() { std::cout << "A::f()" << std::endl; }
9            void g() { std::cout << "A::g()" << std::endl; }
10 private:
11   std::string name;
12 };
13 class B : public A {
14 public:
15   B(const std::string& n, const char* t) :
16     A(n),
17     title(new char[strlen(t)+1]) {
18       strcpy(title, t);
19     }
20   ~B() { delete [] title;  std::cout << "derived" << std::endl; }
21   void f() { std::cout << "B::f()" << std::endl; }
22   void g() { std::cout << "B::g()" << std::endl; }
23 private:
24   char* title;
25 };
26 int main() {
27   A* x = new B("Thane", "Whiterun");
28   x->f();
29   x->g();
30   delete x;
31 }
```

6. (20 points) Write two functions for Student: (1) an assignment operator, and Student::setMajor.

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

class Student : public Person {
public:
  Student() : Person(), major(new char[1]) { major[0] = '\0'; }
  Student(const char* n, const char* m) :
    Person(n), major(new char[strlen(m)+1]) {
    strcpy(major, m);
  }
  virtual ~Student() { delete [] major; }
private:
  char* major;
};
```

7. (10 points) The program below stores the first twenty-five *Fibonacci* numbers in a *list*. Write a function eraseOdd that erases all of the odd *Fibonacci* numbers.

```
1  #include <iostream>
2  #include <list>
3  const int MAX = 25;
4
5  void print(const std::list<int> & ml) {
6    std::list<int>::const_iterator ptr = ml.begin();
7    while ( ptr != ml.end() ) {
8      std::cout << *ptr  << ", ";
9      ++ptr;
10   }
11   std::cout << std::endl;
12 }
13
14 int fibonacci(int n) {
15   if ( n <= 2 ) return 1;
16   else return fibonacci(n-1) + fibonacci(n-2);
17 }
18
19 void init(std::list<int> & ml) {
20   for (unsigned int i = 1; i < MAX; ++i) {
21     ml.push_back( fibonacci(i) );
22   }
23 }
24
25 int main() {
26   std::list<int> mylist;
27   init(mylist);
28   print(mylist);
29 }
```

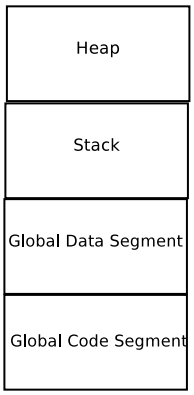

Figure 1: **Memory Layout**.

8. (5 points) Indicate in Figure 1 where a static class variable will be stored.

---

9. (5 points) Fix the syntax error for the program below; it fails to compile, and gives the following error message:

```
main.cpp:5:7: note:   no known conversion for implicit this parameter
  from const Number* to Number*
```

```
1  #include <iostream>
2  class Number {
3  public:
4    Number(int n) : number(n) {}
5    int getNumber() { return number; }
6  private:
7    int number;
8    Number& operator=(const Number&);
9    Number(const Number&);
10 };
11
12 void printNumber(const Number& number) {
13   std::cout << number.getNumber() << std::endl;
14 }
15
16 int main( ) {
17   Number number(17);
18   printNumber( number );
19 }
```

10. (10 points) Convert class Clock to a GoF singleton. Be sure to "Explicitly disallow the use of compiler-generated functions you do not want."

```
1  #include <iostream>
2
3  class Clock {
4  public:
5    Clock() : ticks(0) {}
6    int getTicks() const { return ticks; }
7    void update() { ++ticks; }
8  private:
9    int ticks;
10 };
11
12
13 int main( ) {
14   Clock clock;
15   clock.update();
16   std::cout << clock.getTicks() << std::endl;
17 }
```