1. (20 points) The following program gives the correct output and seems to be a valid program. However, valgrind indicates that 16 bytes are definitely lost and 4 bytes are indirectly lost. (1) Explain where (line numbers) and how the 16 are directly lost and the 4 are indirectly lost. (2) Fix the program so that there are no leaks.

```
  1 #include <iostream>
  2
  3 class B {
  4 public:
  5   B() : number(0) {}
  6 private:
  7   int number;
  8 };
  9
 10 class A{
 11 public:
 12   A() : m(17), n(1), b(new B) {}
 13   int getM() const { return m; }
 14 private:
 15   int m, n;
 16   B* b;
 17   A(const A&);
 18   A& operator=(const A&);
 19 };
 20
 21 int main( ){
 22   A* a = new A;
 23   std::cout << a->getM() << std::endl;
 24 }
 *********************************************************
==5761== HEAP SUMMARY:
==5761==     in use at exit: 20 bytes in 2 blocks
==5761==   total heap usage: 2 allocs, 0 frees, 20 bytes allocated
==5761==
==5761== LEAK SUMMARY:
==5761==    definitely lost: 16 bytes in 1 blocks
==5761==    indirectly lost: 4 bytes in 1 blocks
==5761==      possibly lost: 0 bytes in 0 blocks
==5761==    still reachable: 0 bytes in 0 blocks
==5761==         suppressed: 0 bytes in 0 blocks
```

2. (20 points) The following program uses the Explicit Instantiation Model for templates.

   (a) What is the output of the program?

   (b) Convert it to the Inclusion Model (don't rewrite the code, just circle and use arrows).

```
template <typename TYPE>
class Stack {
public:
  Stack( ) : EMPTY(-1), topOfStack(EMPTY) {};
  void push( TYPE c );
  const TYPE & top() const;
private:
  const int EMPTY;
  enum { BUFFSIZE = 100 };
  TYPE s[BUFFSIZE];
  int topOfStack;
};
*********************************************
#include "stack.h"
template <typename TYPE>
void Stack<TYPE>::push( TYPE c ) {
  s[++topOfStack] = c;
}
template <typename TYPE>
const TYPE & Stack<TYPE>::top() const {
  return s[topOfStack];
}
template class Stack<char>;
template class Stack<float>;
*********************************************
#include  <iostream>
#include  "stack.h"

void convert(int number) {
  Stack<char> stk;
  stk.push( (number % 7) + '0' );
  std::cout << stk.top() << std::endl;
}

int main() {
  convert(25);
  Stack<float> stk;
  stk.push( 25%2?17:33 );
  std::cout << stk.top() << std::endl;
}
```

3. (20 points) Convert class Random to a GoF singleton; be sure to make changes to function main so that it correctly uses the singleton. Make sure you insert code so that your program compiles and links.

```
#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;
}
```

4. (10 points) Give the output for the following program.

```
#include <iostream>
class A {
public:
  A() {}
  virtual void print() const = 0;
  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 print() const { std::cout << "I'm a B" << std::endl; }
  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() {
  A* b = new B;
  b->foo();
  b->bar();
}
```

5. (10 points) Give the output for the following program. There is no destructor in the program. If the program is changed so that one additional push_back is added after line 11, what would the size and capacity be?

```
 1 #include <iostream>
 2 #include <vector>
 3 class A{
 4 public:
 5   A() { std::cout << "construct" << std::endl; }
 6   A(const A&) { std::cout << "copy" << std::endl; }
 7 };
 8 int main() {
 9   std::vector<A> a;
10   a.push_back(A());
11   a.push_back(A());
12   std::cout << a.size() << std::endl;
13   std::cout << a.capacity() << std::endl;
14 }
```

6. (20 points) There are 9 classes in the basic animation framework that you used for the first project, and two of these classes are Manager and Frame.

   (a) What are the other 7 classes.

   (b) Why is Frame a Flyweight, and how does the Manager class participate in this pattern?