1. (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("cat"), y = x;
}
```

****

```
convert
copy
destructor
destructor
```

---

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

```
#include <iostream>
class Bird {
public:
  Bird(int w) : wingSpan(w), speed(2*wingSpan) {
    std::cout << "Speed: " << speed << std::endl;
    std::cout << "Wing Span: " << wingSpan << std::endl;
  }
private:
  int speed;
  int wingSpan;
};

int main() {
  Bird robin(27);
}
```

```
Speed: 0
Wing Span: 27
```

3. (20 points) Give the output for the following program:

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

class Student {
public:
  Student(float g) {
    gpa = g;
  }
private:
  Number gpa;
};

int main() {
  Student* npc = new Student(3.4);
}
```

```
default
convert
assign
destructor
```

4. (20 points) Give the output for the following program:

```
#include <iostream>
#include <cstring>
#include <string>
class A {
public:
  A(const std::string& n) : name(n) {}
  ~A() { std::cout << "base" << std::endl; }
  virtual void f() { std::cout << "A::f()" << std::endl; }
          void g() { std::cout << "A::g()" << std::endl; }
private:
  std::string name;
};
class B : public A {
public:
  B(const std::string& n, const char* t) :
    A(n),
     title(new char[strlen(t)+1]) {
       strcpy(title, t);
     }
  ~B() { delete [] title;  std::cout << "derived" << std::endl; }
  void f() { std::cout << "B::f()" << std::endl; }
  void g() { std::cout << "B::g()" << std::endl; }
private:
  char* title;
};
int main() {
  A* x = new B("Thane", "Whiterun");
  x->f();
  x->g();
  delete x;
}
```

```
B::f()
A::g()
base
```

5. (20 points) Give the output for the following program:

```
#include <iostream>
#include <vector>
const int MAX = 2;

class Number {
public:
  Number() : number(0) { std::cout << "default" << std::endl; }
  explicit Number(int n) : number(n) {
    std::cout << "convert: " << n << std::endl;
  }
  Number(const Number& a) : number(a.number) {
    std::cout << "copy: " << a.number << std::endl;
  }
  Number& operator=(const Number& rhs) {
    number = rhs.number;
    std::cout << "assign" << std::endl;
    return *this;
  }
  int getNumber() const { return number; }
private:
  int number;
};

void print(const std::vector<Number> & vec) {
  for (unsigned int i = 0; i < vec.size(); ++i) {
    std::cout << vec[i].getNumber()  << ", ";
  }
  std::cout << std::endl;
}

void init(std::vector<Number> & vec) {
  for (unsigned int i = 0; i < MAX; ++i) {
    vec.push_back( Number(i+1) );
  }
}

int main() {
  std::vector<Number> vec;
  vec.reserve(2);
  init(vec);
  vec.push_back( Number(99) );
  std::cout << "SIZE: " << vec.size() <<  std::endl;
  std::cout << "CAP:  " << vec.capacity() <<  std::endl;
  print(vec);
}
```

```
convert: 1
copy: 1
convert: 2
copy: 2
convert: 99
copy: 99
copy: 1
copy: 2
SIZE: 3
CAP:  4
1, 2, 99,
```

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

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

class Person {
public:
  Person() : name(new char[1]) { name[0] = '\0'; }
  Person(const char* b) : name(new char[strlen(b)+1]) {
    strcpy(name, b);
  }
  Person(const Person& s) : name(new char[strlen(s.name)+1]) {
    strcpy(name, s.name);
  }
  virtual ~Person() { delete [] name; }
  const char* getName() const { return name; }
  Person operator+(const Person&) const;
  Person& operator=(const Person& rhs) {
    if ( this == &rhs ) return *this;
    delete [] name;
    name = new char[strlen(rhs.name)+1];
    strcpy(name, rhs.name);
    return *this;
  }
  virtual void display() const { std::cout << name; }
private:
  char* name;
};

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; }
  Student& operator=(const Student& rhs) {
    if ( this == &rhs ) return *this;
    Person::operator=(rhs);
    setMajor(rhs.major);
    return *this;
  }
  void setMajor(const char* m) {
    delete [] major;
    major = new char[strlen(m)+1];
    strcpy(major, m);
  }
  virtual void display() const {
    Person::display();
    std::cout << ",␣" << major;
  }
private:
  char* major;
};

void display( const std::vector<Person*> & people ) {
```

```cpp
  for (unsigned int i = 0; i < people.size(); ++i) {
    people[i]->display();
    std::cout << std::endl;
  }
}

void destroy( std::vector<Person*> & people ) {
  for (unsigned int i = 0; i < people.size(); ++i) {
    delete people[i];
  }
}


int main( ) {
  std::vector<Person*> people;
  people.push_back(new Student("Peter␣Quill", "Guardian"));
  people.push_back(new Person("Peter␣Parker"));
  static_cast<Student*>(people[0])->setMajor("Guardian␣of␣Universe");
  display(people);
  destroy(people);
}
```