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

```cpp
#include <iostream>
int main() {
  int a = 9, b = 11;
  int c = a%3?b:17;
  std::cout << c << std::endl;
  int v = 320;
  v = v*((a%2)?-1:1);
  std::cout << v << std::endl;
}
```

---

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

```cpp
#include <iostream>
#include <cstdlib>
#include <vector>
const int MAX = 1000;

int main() {
  std::vector<int> vec;
  for (int i = 0; i < MAX; ++i) {
    vec.push_back( rand()%100 );
  }
  vec.push_back(2);
  std::cout << vec.size() << std::endl;
  std::cout << vec.capacity() << std::endl;
}
```

---

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

```cpp
#include <string>
#include <vector>
#include <iostream>
class Pokemon {
public:
  Pokemon( ) : name() { std::cout << "default" << std::endl; }
  Pokemon(const std::string& n) : name(n) {
    std::cout << "conversion" << std::endl;
  }
  Pokemon(const Pokemon& p ) : name(p.name) {
    std::cout << "copy" << std::endl;
  }
private:
  std::string name;
};
int main() {
  std::vector<Pokemon> pokes;
  pokes.push_back(std::string("Larvitar"));
  pokes.push_back(Pokemon("Steelix"));
}
```

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

```
#include <string>
#include <vector>
#include <iostream>
class Pokemon {
public:
  Pokemon( ) : name() { std::cout << "default" << std::endl; }
  Pokemon(const std::string& n) : name(n) {
    std::cout << "conversion" << std::endl;
  }
  Pokemon(const Pokemon& p ) : name(p.name) {
    std::cout << "copy" << std::endl;
  }
private:
  std::string name;
};
int main() {
  std::vector<Pokemon> pokes;
  pokes.reserve(2);
  pokes.push_back(std::string("Larvitar"));
  pokes.push_back(Pokemon("Steelix"));
}
```

---

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

```
#include <string>
#include <vector>
#include <iostream>
class Pokemon {
public:
  Pokemon( ) : name() { std::cout << "default" << std::endl; }
  Pokemon(const std::string& n) : name(n) {
    std::cout << "conversion" << std::endl;
  }
  Pokemon(const Pokemon& p ) : name(p.name) {
    std::cout << "copy" << std::endl;
  }
private:
  std::string name;
};
int main() {
  std::vector<Pokemon> pokes;
  pokes.reserve(2);
  pokes.emplace_back(std::string("Larvitar"));
  pokes.emplace_back(std::string("Steelix"));
}
```

6. (10 points)

   (a) Give the output for the following program.

   (b) Without changing the functionality, make the program more efficient by preferring initialization to assignment?

```
#include <string>
#include <vector>
#include <iostream>
class Pokemon {
public:
  Pokemon( ) : name() { std::cout << "default" << std::endl; }
  Pokemon(const std::string& n) : name(n) {
    std::cout << "conversion" << std::endl;
  }
  Pokemon(const Pokemon& p ) : name(p.name) {
    std::cout << "copy" << std::endl;
  }
  Pokemon& operator=(const Pokemon&) {
    std::cout << "assignment" << std::endl;
    return *this;
  }
private:
  std::string name;
};

class Pokedex {
public:
  Pokedex(const std::string& s) {
    pokemon = s;
  }
private:
  Pokemon pokemon;
};

int main() {
  Pokedex p("Steelix");
}
```

7. (15 points) For the following program:

   (a) Write a default constructor.

   (b) Write function isUnique, which returns true if n is not in names, and false otherwise.

   (c) Write function display, which lists the items in names.

```
#include <string>
#include <vector>
#include <algorithm>
#include <iostream>

class Pokedex {
public:
  void add(const std::string& n) {
    if ( isUnique(n) ) names.push_back(n);
  }

  void display() const;

private:
  std::vector< std::string > names;
  bool isUnique( const std::string& ) const;
};

int main() {
  Pokedex pokedex;
  pokedex.add("Steelix");
  pokedex.display();
}
```

8. (15 points) Convert class Clock to a singleton. Be sure to "Explicitly disallow the use of compiler-generated functions you do not want."

```
#include <iostream>

class Clock {
public:
  Clock() : ticks(0) {}
  int getTicks() const { return ticks; }
  void update() { ++ticks; }
private:
  int ticks;
};


int main( ) {
  Clock clock;
  clock.update();
  std::cout << clock.getTicks() << std::endl;
}
```

9. (10 points) Write an assignment operator for class string.

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

class string {
public:
  string() : buf(new char[1]) { buf[0] = '\0'; }
  string(const char* b) : buf(new char[strlen(b)+1]) {
    strcpy(buf, b);
  }
  ~string() { delete [] buf; }
  const char& operator[](int index) const { return buf[index]; }
  char& operator[](int index) { return buf[index]; }
private:
  char* buf;
};

int main() {
  string a("dog"), b;
  b = a;
  std::cout << b[0] << std::endl;
}
```