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

```
#include <iostream>
int cout = 45%7?5:0;
int endl = 7/2;
namespace Mine {
  namespace std {
    float cout = 2/4;
  }
}
int main() {
  std::cout << cout+endl << std::endl;
  std::cout << Mine::std::cout << std::endl;
}
```

```
8
0
```

---

2. (15 points) Give the output for the following program.

```
#include <iostream>
class Pokemon {
public:
  Pokemon() { std::cout << "default" << std::endl; }
  Pokemon(int) { std::cout << "convert" << std::endl; }
  Pokemon(const Pokemon&) { std::cout << "copy" << std::endl; }
  Pokemon& operator=(const Pokemon&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
private:
  int cp;
};
class Pokedex {
public:
  Pokedex(const Pokemon& p) { mon = p; }
private:
  Pokemon mon;
};
int main() {
  Pokedex dex(2750);
}
```

```
convert
default
assign
```

---

3. (5 points) Write a C++ program that generates and prints a random number between 5 and 10, including the end points.

```
#include <iostream>
int main() {
  std::cout << rand()%6+5<< std::endl;
}
```

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

```
#include <iostream>
#include <vector>
class Pokemon {
public:
  Pokemon() { std::cout << "default" << std::endl; }
  Pokemon(int) { std::cout << "convert" << std::endl; }
  Pokemon(const Pokemon&) { std::cout << "copy" << std::endl; }
  Pokemon& operator=(const Pokemon&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
private:
  int cp;
};
int main() {
  std::vector<Pokemon> vec;
  for (unsigned int i = 0; i < 2; ++i) {
    vec.push_back( i );
  }
}
```

```
convert
copy
convert
copy
copy
```

---

5. (10 points) Give the output for the following program. Note the use of reserve.

```
#include <iostream>
#include <vector>
class Pokemon {
public:
  Pokemon() { std::cout << "default" << std::endl; }
  Pokemon(int) { std::cout << "convert" << std::endl; }
  Pokemon(const Pokemon&) { std::cout << "copy" << std::endl; }
  Pokemon& operator=(const Pokemon&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
private:
  int cp;
};
int main() {
  std::vector<Pokemon> vec;
  vec.reserve(2);
  for (unsigned int i = 0; i < 2; ++i) {
    vec.push_back( i );
  }
}
```

```
convert
copy
convert
copy
```

6. (15 points) Give the output for the following program. After printing the output the program crashes with a double free error. To fix the error, what function should the programmer write?

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

class string {
public:
  string(const char * s) : buf(new char[strlen(s)+1]) {
    strcpy(buf, s);
    std::cout << "convert" << std::endl;
  }
  ~string() {
    delete [] buf;
    std::cout << "destructor" << std::endl;
  }
  char* getBuf() const { return buf; }
  string& operator=(const string&) {
    std::cout << "assign" << std::endl;
    return *this;
  }
private:
  char *buf;
};

int main() {
  string x("cat"), y = x;
  char* buf = x.getBuf();
  buf[0] = 'r';
  std::cout << x.getBuf() << std::endl;
}
```

```
convert
rat
destructor
*** Error in `./run': double free or corruption
Need a copy constructor
```

---

7. (5 points) Give the output for the following program.

```
#include <iostream>
class Pokemon {
public:
  Pokemon(int p) : cp(p) { std::cout << "convert" << std::endl; }
  Pokemon& operator=(const Pokemon& p) {
    cp = p.cp;
    std::cout << "assign" << std::endl;
    return *this;
  }
private:
  int cp;
};
int main() {
  Pokemon p(2500), q = p;
}
```

```
convert
```

8. (30 points) Write a copy constructor, assignment operator, and an output operator for the following program.

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

class Pokemon {
public:
  Pokemon(const char* n) : name(new char[strlen(n)+1]) {
    strcpy(name, n);
  }
  Pokemon(const Pokemon& p) : name(new char[strlen(p.name)+1]) {
    strcpy(name, p.name);
  }
  Pokemon& operator=(const Pokemon& rhs) {
    if ( this == &rhs ) return *this;
    delete [] name;
    name = new char[strlen(rhs.name)+1];
    strcpy(name, rhs.name);
    return *this;
  }
  const char* getName() const { return name; }
private:
  char* name;
};
std::ostream& operator<<(std::ostream& out, const Pokemon& p) {
  return out << p.getName();
}

int main() {
  Pokemon p("snorlax");
  std::cout << p << std::endl;
}
```