1. (15 points) For the following program, write function removeMultiples, used on line 26, so that it removes all multiples of mult from mylist.

```
1  #include <iostream>
2  #include <list>
3  #include <cstdlib>
4  #include <algorithm>
5  const int MAX = 20;
6  const int MAX_NUMBER = 100;
7
8  void removeMultiples(std::list<int> & mylist, int multiple) {
9    std::list<int>::iterator it = mylist.begin();
10   while (it != mylist.end()) {
11     if (*it%multiple == 0) {
12       it = mylist.erase(it);
13     }
14     else ++it;
15   }
16 }
17
18 void init(std::list<int> & mylist) {
19   for (unsigned int i = 0; i < MAX; ++i) {
20     mylist.push_back( rand() % MAX_NUMBER );
21   }
22 }
23
24 void print(const std::list<int> & mylist) {
25   std::list<int>::const_iterator ptr = mylist.begin();
26   while ( ptr != mylist.end() ) {
27     std::cout << (*ptr)  << ", ";
28     ++ptr;
29   }
30   std::cout << std::endl;
31 }
32
33 int main() {
34   std::list<int> mylist;
35   init(mylist);
36   print(mylist);
37   int multiple = rand()%3+2;
38   removeMultiples(mylist, multiple);
39   std::cout << "List with multiples of " << multiple
40             << " removed" << std::endl;
41   print(mylist);
42 }
```

2. (15 points) For the following program, write the missing function, removeOddAfterOdd, which removes all odd numbers that follow an odd number. Sample output might be:

```
89, 6, 12, 20, 52, 3, 71, 0, 7, 93, 79, 61, 85, 36, 86, 7, 23, 83, 68, 80,
List with odd after odd removed:
89, 6, 12, 20, 52, 3, 0, 7, 36, 86, 7, 68, 80,
```

```
1  #include <iostream>
2  #include <list>
3  #include <algorithm>
4  #include <cstdlib>
5  #include <ctime>
6  const int MAX = 20;
7  const int MAX_NUMBER = 100;
8
9  void removeOddAfterOdd(std::list<int> & mylist) {
10   std::list<int>::iterator it = mylist.begin();
11   std::list<int>::iterator next = ++it;
12   while (next != mylist.end()) {
13     if (next!=mylist.end() && *it%2==1 && *next%2==1) {
14       next = mylist.erase(next);
15     }
16     else {
17       it = next;
18       ++next;
19     }
20   }
21 }
22
23 void init(std::list<int> & mylist) {
24   for (unsigned int i = 0; i < MAX; ++i) {
25     mylist.push_back( rand() % MAX_NUMBER );
26   }
27 }
28
29 void print(const std::list<int> & mylist) {
30   for ( int n : mylist ) {
31     std::cout << n  << ", ";
32   }
33   std::cout << std::endl;
34 }
35
36 int main() {
37   srand( time(0) );
38   std::list<int> mylist;
39   init(mylist);
40   print(mylist);
41   removeOddAfterOdd(mylist);
42   std::cout << "List with odd after odd removed: " << std::endl;
43   print(mylist);
44 }
```

3. (10 points) Add code so that when the list is printed on line 23 the numbers are sorted (low to high).

```
1  #include <iostream>
2  #include <list>
3  #include <cstdlib>
4  #include <algorithm>
5  const int MAX = 20;
6  const int MAX_NUMBER = 100;
7
8  void init(std::list<int> & mylist) {
9    for (unsigned int i = 0; i < MAX; ++i) {
10     mylist.push_back( rand() % MAX_NUMBER );
11   }
12 }
13
14 void print(const std::list<int> & mylist) {
15   std::list<int>::const_iterator ptr = mylist.begin();
16   while ( ptr != mylist.end() ) {
17     std::cout << (*ptr) << ", ";
18     ++ptr;
19   }
20   std::cout << std::endl;
21 }
22
23 int main() {
24   std::list<int> mylist;
25   init(mylist);
26   print(mylist);
27   mylist.sort();
28   print(mylist);
29 }
```

---

4. (25 points) Write a function object, used on line 33, and an overloaded output operator, used on line 26, so that the numbers are sorted (low to high).

```
1  #include <iostream>
2  #include <list>
3  #include <cstdlib>
4  #include <algorithm>
5  const int MAX = 20;
6  const int MAX_NUMBER = 100;
7
8  class Number {
9  public:
10   Number() : number(0) { }
11   Number(int n) : number(n) {
12   }
13   Number(const Number& a) : number(a.number) { }
14   int getNumber() const { return number; }
15   bool operator<(const Number& rhs) const { return number < rhs.number; }
16 private:
17   int number;
18 };
19 std::ostream& operator<<(std::ostream& out, const Number* number) {
20   return out << number->getNumber();
21 }
22
23 class NumberLess{
24 public:
25   bool operator()(const Number* lhs, const Number* rhs) const {
26     return lhs->getNumber() < rhs->getNumber();
27   }
28 };
29
30 void init(std::list<Number*> & numberList) {
31   for (unsigned int i = 0; i < MAX; ++i) {
32     numberList.push_back( new Number(rand() % MAX_NUMBER) );
33   }
34 }
35
36 void print(const std::list<Number*> & numberList) {
37   std::list<Number*>::const_iterator ptr = numberList.begin();
38   while ( ptr != numberList.end() ) {
39     std::cout << (*ptr) << ", ";
40     ++ptr;
41   }
42   std::cout << std::endl;
43 }
44
45
46 int main() {
47   std::list<Number*> numberList;
48   init(numberList);
49   print(numberList);
50   numberList.sort(NumberLess());
51   print(numberList);
52 }
```

5. (15 points) Write function nameFound, used on line 28, so that it returns *true* if names[index] is found in mymap, false otherwise.

```
1  #include <iostream>
2  #include <vector>
3  #include <map>
4  #include <cstdlib>
5  #include <ctime>
6  #include <algorithm>
7
8  void init(std::map<std::string, int> & mymap, const
9            std::vector<std::string>& names) {
10   for (const auto& n : names) {
11     mymap[n] = rand() % 100;
12   }
13 }
14
15 void print(const std::map<std::string, int>& mymap) {
16   for (const auto& n : mymap) {
17     std::cout << n.first << ", " <<  n.second << std::endl;
18   }
19 }
20
21 bool nameFound(const std::map<std::string, int>& mymap,
22                std::vector<std::string>& names, int index) {
23   std::string searchString = names[index];
24   return mymap.find(names[index]) != mymap.end();
25 }
26
27 int main() {
28   srand( time(0) );
29   std::vector<std::string> names = {"Red Oak", "Sugar Maple", "Chestnut"};
30   std::map<std::string, int> mymap;
31   init(mymap, names);
32   names.push_back("Ash");
33   names.push_back("Cedar");
34   names.push_back("Elm");
35   print(mymap);
36
37   int index = rand()%names.size();
38
39   if ( nameFound(mymap, names, index) ) {
40     std::cout << names[index] << " found" << std::endl;
41   }
42   else {
43     std::cout << names[index] << " not found" << std::endl;
44   }
45 }
```

6. (20 points) For the following program:

   (a) Give the output, and

   (b) There are seven (7) uses of const missing from the first 21 lines of code in the following program. Supply these missing uses of const.

```
1  #include <iostream>
2  #include <cstring>
3
4  class string {
5  public:
6    string(const char *b) : buf(new char[strlen(b)+1]) {
7      strcpy(buf, b);
8    }
9    string(const string &rhs) : buf(new char[strlen(rhs.buf)+1]) {
10     strcpy(buf, rhs.buf);
11   }
12   ~string() { delete [] buf; }
13
14   const char* getBuf() const { return buf; }
15
16   void setBuf(const char* b);
17   size_t size() const { return strlen(buf); }
18   const char& operator[](int index) const {
19     std::cout << "const []" << std::endl;
20     return buf[index];
21   }
22   char& operator[](int index) {
23     std::cout << "non const []" << std::endl;
24     return buf[index];
25   }
26   string& operator=(const string& rhs) = delete;
27 private:
28   char *buf;
29 };
30
31 int main() {
32   string s1 = "Hello";
33   std::cout << s1[0] << std::endl;
34   s1[0] = 'x';
35 }
```