1. (15 points) The following program generates a positive random number and then uses the C++ ternary operator to possibly make the number negative. Refactor the program so that it uses a *lambda* function rather than the ternary operator to possibly **make** the number negative.

```
#include <iostream>
#include <ctime>
// [capture clause] (parameters) -> return-type {body}

int main() {
  srand( time(0) );
  int x = rand() % 100;

  std::cout << "x before " << x << std::endl;
  auto maybe = [](int& x){ if (rand()%2) return x=-x; else return x; };
  std::cout << "x " << maybe(x) << std::endl;
  std::cout << "x after " << x << std::endl;
}
```

---

2. (15 points) The following program reads a character from the user and then prints the character. Extend the program so that it uses a lambda function to determine if the character is an upper case letter and then prints an appropriate message. (You may not use isalpha, or any other built-in function). Possible output:

```
A
is: 1
malloy@aramis:~/pubgit/4160-2017/quiz/3/code/isLetter/soln$ r
8
is: 0
```

```
#include <iostream>
#include <ctime>
// [capture clause] (parameters) -> return-type {body}

int main() {
  char ch;
  std::cin >> ch;
  std::cout << ch << std::endl;
  //bool flag = isalpha(ch);
  //if ( flag )
    //std::cout << "letter" << std::endl;
  //else
    //std::cout << "NOT letter" << std::endl;
  auto isletter = [](char ch) { return (ch >= 'A' && ch <= 'Z'); };
  std::cout << "is: " << isletter(ch) << std::endl;

}
```

3. (15 points) Give the output for the following program, and then write function display, which prints the key and value for each item in a map. (You can use a while loop or a ranged for loop)

```
#include <iostream>
#include <string>
#include <map>

void display( const std::map<std::string, int>& pokemon ) {
  for ( auto it : pokemon ) {
    std::cout << it.first << ", " << it.second << std::endl;
  }
}

int main() {
  std::map<std::string, int> pokemon;
  pokemon["Snorlax"] = 2750;
  pokemon["Pikachu"] = 1725;
  pokemon["Snorlax"] = 1750;
  std::cout << pokemon.size() << std::endl;
  display( pokemon );
}
```

---

4. (15 points) Using an *iterator* or *const_iterator* (you decide) and a **while** loop, unroll the ranged for loop in display. You may **not** use auto.

```
#include <iostream>
#include <list>

void display(const std::list<int>& numbers) {
  std::list<int>::const_iterator it = numbers.begin();
  while ( it != numbers.end() ) {
    std::cout << *it << std::endl;
    ++it;
  }
}

int main() {
  std::list<int> numbers;
  numbers.push_back( rand() % 100 );
  numbers.push_back( rand() % 100 );
  numbers.push_back( rand() % 100 );
  numbers.push_back( rand() % 100 );
  display(numbers);
}
```

5. (40 points) Extend the program below by adding:

   - (5 pts) An output operator for Number;
   - (15 pts) Write a function object and use it to sort numberList;
   - (10 pts) Write a lambda function to replace the function object;
   - (10 pts) Generate a random number in the range 0 to MAX and write code to search for the random number and print a message indicating whether or not the random number was found.

```
#include <iostream>
#include <ctime>
#include <list>
#include <algorithm>
const int MAX = 20;

class Number {
public:
  Number() : number(0) { }
  Number(int n) : number(n) { }
  Number(const Number& a) : number(a.number) { }
  int getNumber() const { return number; }
  bool operator<(const Number& rhs) const { return number < rhs.number; }
private:
  int number;
};
std::ostream& operator<<(std::ostream& out, const Number* number) {
  return out << number->getNumber();
}

struct LessThan {
  bool operator()(const Number* rhs, const Number* lhs) const {
    return rhs->getNumber() < lhs->getNumber();
  }
};

class Target {
public:
  Target( int n ) : number(n) {}
  bool operator()(const Number* rhs) const {
    return number == rhs->getNumber();
  }
private:
  int  number;
};

void init(std::list<Number*> & numberList) {
  for (unsigned int i = 0; i < MAX; ++i) {
    numberList.push_back( new Number(rand()%MAX) );
  }
}

void print(const std::list<Number*> & numberList) {
  for ( Number* n : numberList ) {
    std::cout << n  << ", ";
  }
  std::cout << std::endl;
}
```

```
49
50 int main() {
51   srand( time(0) );
52   std::list<Number*> numberList;
53   init(numberList);
54   print(numberList);
55   numberList.sort(LessThan());
56   print(numberList);
57
58   std::list<Number*>::iterator it =
59     find_if( numberList.begin(), numberList.end(), Target(75) );
60   if (  it == numberList.end() ) std::cout << "NO" << std::endl;
61   else std::cout << "YES" << std::endl;
62
63   int number = rand()%MAX;
64   auto f = [&number](const Number* n) { return n->getNumber()==number; };
65   it = find_if( numberList.begin(), numberList.end(), f );
66   if (  it == numberList.end() ) {
67     std::cout << number << " not found" << std::endl;
68   }
69   else std::cout << number << " found" << std::endl;
70 }
```