Fall 2008 EE 380L Midterm: NAME: _____

1. The purpose of this question is to evaluate whether you understand the principles of writing generic functions in the STL. To answer this question, you must **write a template function to compute the dot product of two vectors**. You may assume:
   - Both of the vectors have the same element type. The element type is some sort of numeric type (something that can be added with "+" and multiplied with "*").
   - Both vectors have the same number of elements
   - The vectors are not necessarily stored in the std::vector<T> data structure. In fact, the vectors may be in any data structure (linked lists, ordinary arrays, etc.).

   Recall that the "dot product" for two vectors x, and y can be computed using (if x and y were ordinary arrays of length N)

```
s = 0;
for (int k = 0; k < N; k += 1) {
     s += x[k] * y[k];
}
return s;
```

   Your function should have three arguments. The first two arguments are used to describe the first vector using the conventions of the standard C++ library. The final argument is an iterator that points to the beginning of the second vector (recall, both vectors are known to have the same number of elements). Your function should be called "dotProd". Partial credit will be awarded as follows:

   a. (10 points) Declare the template correctly. Recognize that declaring the return value is a bit tricky.
   b. (10 pts) Write the method correctly. Use only operators and functions appropriate for all data structures that may be used to store the vectors.

2. (15 points) Please rewrite question 1 so that five arguments are passed. The first three arguments remain the same as for question 1. The fourth argument is a function that can be used in lieu of the “multiplication” operation for dot product. The second argument should be a function that can be used in lieu of the “addition” operation. Part of the question is to know the convention for how “functions” are passed as parameters to generic functions in the STL.

3. Design and implement a class hierarchy for a *Debater* as described in further detail below. Note that your solution will involve several classes.

   A Debater has three methods (*foreign*, *domestic*, and *energy*). Each method takes no argument and returns a string. I would like to be able to use the following function, precisely as written, with your Debater implementation. Please ensure your solution works

```
void debate(vector<Debater> candidates) {
  int first_candidate = drand48() * candidates.size();
  int k = first_candidate;
  do {
    cout << "what is your opinion on foreign policy?\n";
    cout << candidates[k].foreign();
    cout << "what is your opinion on domestic policy?\n";
    cout << candidates[k].domestic();
    cout << "what is your opinion on energy policy?\n";
    cout << candidates[k].energy();
    k = (k + 1) % candidates.size();
  } while (k != first_candidate);
}
```

   You must design an Object Oriented solution, with no if statements, switch statements, look-up tables or type casts. You may use as many virtual functions as you need.

   There are three types of Debaters:

   - Republican – a republican responds with "bomb them" for foreign policy, with "let them eat cake" for domestic policy, and with "drill now" for energy policy.
   - Democrat – a democrat responds with "what policy?" for foreign policy, with "tax the rich" for domestic policy, and with "flower power" for energy policy.
   - Moderate – a moderate responds with precisely the same response that whatever the previous Debater gave to the same question. If a moderate is the first Debater to answer some question, then the moderate responds, "a chicken in every pot" regardless of the question. HINT: you may want to use global variables or static variables in some way to implement your solution.

Partial credit will be assigned as follows – note, if your solution is not "object oriented" you'll receive ZERO POINTS for this problem. You have two pages for your response. Please don't feel that you need to use all of that space.

a. (10 points) You do not need to write a "factory" for Debaters. You do not need to solve the problem of how a vector<Debater> is populated with different types of Debaters. You do need to make sure that the function works as written – NOTE: the argument to the function is passed by value, not by reference.
b. (15 points) Debaters must behave as required, 5 points for each Debater type
c. (5 points) other syntax and semantic requirements

Define your base class(es) on this page

Put your implementation of Republican, Democrat and Moderate on this page

4. (5 points) Write a template function called “has_virtual” that takes an argument of arbitrary type T and returns true if T has at least one virtual function, and returns false if T has no virtual functions. Note that T could be a base type (e.g., **int**). HINT: You might want to create a template class too, and you might want to use **sizeof**.

5. Normally, we think of data structures as containers that we place data into and take data out of. However, other things can be containers, and iterators can be used to access their "contents". For example, the set of all integers is a container and can be accessed with an iterator like the following:

```
class IntegerRange {
public:
   class const_iterator {
     int v;
   public:
     const_iterator(int v) { this->v = v; }
     int operator*() { return v; }
     const_iterator& operator++(void) {
       v += 1; return *this;
     }
     bool operator==(const iterator& that) {
       return v == that.v;
     }
   };
   const_iterator begin(void) {
      return const_iterator(start); }
   const_iterator end(void) {
      return const_iterator(last+1); }
   IntegerRange(s, l) { start = s; last = l; }
};
int main(void) {
  IntegerRange r(1, 10);
  IntegerRange::const_iterator p = r.begin();
  while (p != r.end()) {
    cout << *p << endl;
    ++p;
  }
}
```

a. (5 points) What iterator category is appropriate for this iterator the way it is written? Explain.

b. (5 points) What's missing from the iterator class (methods, nested types, data members, constructors, etc)?

c. (10 points) Design a similar iterator that returns the values that are Fibonacci numbers within an integer range. Recall that the Fibonacci numbers are 0, 1 and any number that is the sum of two successive Fibonacci numbers (i.e., 0, 1, 1, 2, 3, 5, 8, 13, 21…). Use an FibRange "container" class similar to my IntegerRange example. Your solution must have O(1) space complexity (i.e., no arrays or other ordinary data structures in your solution).

6. (10 pts) I have the following classes:

```
class Base {
public:
  virtual void prompt(void) = 0;
  void doit(void) {
    prompt();
  }
};
class Polite : public Base {
public:
  virtual void prompt(void) {
    cout << "please: ";
  }
};
class Rude : public Base {
public:
  virtual void prompt(void) {
    cout << "gimme a number: ";
  }
};
class VeryRude : public Polite {
public:
  virtual void prompt(void) {
    cout << "gimme a number, dammit: ";
  }
  void doit(void) {
    Base::doit();
  }
}
```

What happens when the following code is run (indicate the output on the lines provided).

```
int main(void) {
  Polite p;
  Rude r;
  VeryRude vr;
  Base* b;
  p.doit();       a._____________________________

  r.doit();       b._____________________________

  vr.doit();      c._____________________________

  b = &vr;
  b->doit();      d._____________________________
}
```