Recall that in the first lab assignment you are required to create at least two classes. Briefly describe the classes you chose to implement. (For each class you decided to implement: give its name, purpose, and how it helps to solve the decyrption task.)

Future lab assignments will have you working with a partner. If you care, please provide lists of names for:

Prefer Prefer **not**

Give the asymptotic time complexity for the following function:

```
int quiz3(int i, int j) {
  int retVal = 0;
  while (i != j) {
    if(i<j) {
      j--;
    } else {
      retVal = quiz3(i,j+1);
    }
  }
  return retVal;
}
```

Using inheritance, write the header file for an adaptor class for the stack data structure that uses a list of ints for the base class.

Write a generic algorithm called encrypt that is passed two iterators, start and stop. The algorithm should encrypt every element in the container between start and stop (including start but not stop). You may assume that a templated function, void scramble(T& element) is available to you. This function scramble function encrypts the element that is passed to it. **What kind of iterator is required by your algorithm?**

Pick two data structures from the STL other than the vector and list and describe them. Give enough detail to convince me that you understand how they are different from the other data structures in the STL.

Suppose you have the following class:

```
class Dictionary {
public:
  ...
  // Reads all of the words in the input stream (is) and adds them to
  //  the dictionary object.
  bool read(istream& is);
  ...
private:
  set<string> words;         // contains all the words in the dictionary
  unsigned int longestWord; // length of the longest word in the dictionary
};
```

Implement the read member function so that it is as efficient as possible. You may assume that we will use the dictionary file from lab 1.

Give three unique characteristics of a **good** hashing function.

Implement the version of erase member function found in the class definition below for the BinTree that we have been developing in class. Note this function does not return anything. **You may assume that the node you are erasing is a leaf node.**

```
template <class T, class unary>
class BinTree {
public:
  BinTree();
  BinTree(const BinTree<T, unary>& org);
  ~BinTree();
  BinTree<T, unary>& operator=(const BinTree<T, unary>& rhs);
  unsigned int size() const;
  ...
  void erase(const T& val);
  ...

private:
  Node<T>* root;
  unsigned int numNodes;
};
```

Implement the version of insert member function found in the class definition below for the BinTree that we have been developing in class. Note this function does not return anything.

```
template <class T, class unary>
class BinTree {
public:
  BinTree();
  BinTree(const BinTree<T, unary>& org);
  ~BinTree();
  BinTree<T, unary>& operator=(const BinTree<T, unary>& rhs);
  unsigned int size() const;
  ...
  void insert(const T& val);
  ...

private:
  Node<T>* root;
  unsigned int numNodes;
};
```

Justify the metric that you used for rating each hash table configuration in lab 5.