(a) What does the following do?

```
std::vector<double> vec(312, 17);
```

(b) Suppose you have a templated vector class with the following data members:

```
T* elements;
long unsigned int capacity;
long unsigned int numberOfElements;
```

Implement the two parameter constructor that functions like the std::vector. The capacity should be the same as the size of the vector unless the size is less than 16, in which case the capacity should be set to 16.

Give the worst case asymptotic growth rate using big-oh notation for the following function:

```
double quiz3(const std::vector<double>& vec)
{
  int randNumsToGenerate = vec.size()/3;
  srand(time(0));
  int index = rand()%vec.size();
  for(unsigned int i=0; i<randNumsToGenerate; ++i) {
    int temp = rand()%vec.size();
    if(index>temp) {
      index = temp;
    }
  }
  return vec[index];
}
```

Implement the stack<T, CONTAINER>::empty() member function. Your implementation should assume that the stack class has the following data member:

```
private:
  CONTAINER data;
```

Write a program that will display each command line argument that begins with a number. For example,

```
>quiz5.exe monkeys eat 7 beats with 13peanuts
```

Should result in the following output:

```
7
13peanuts
```

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

using namespace std;

int main(int argc, char** argv)
{
```

```
  return EXIT_SUCCESS;
}
```

Complete the following templated function which returns `true` if and only if target is found in data.

```
template <class T>
bool isInTheSet(const std::set<T>& data, const T& target) {
  bool found = false;
```

```
  return found;
}
```

Suppose the HashTable class discussed in lecture is modified so that it looks like this:

```
template <class T, class HashFun>
class HashTable {
public:‘\ctp‘
  HashTable(unsigned int tblSize) : buckets(tblSize) { }
  ~HashTable();
  bool empty() const;
  void insert(const T& val);
  bool find(const T& val) const;
protected:
  std::vector<std::list<T> > buckets;
  HashFun hash;
};
```

Implement the find member function for **this** HashTable class.

In lab 5 you were given an abstract Dictionary class. Assume that you have implemented VectorDictionary and SortedVectorDictionary classes that inherit the Dictionary class. Write a program that asks the user to choose which dictionary they would like to use, and then creates the appropriate dictionary object, loads the dictionary with words from the file "words.txt" and tries to find the word "freedom" in the dictionary, and displays an appropriate message. The Dictionary class is given on the back of this page.

```
#ifndef DICTIONARY_H
#define DICTIONARY_H

#include <string>

class Dictionary
{
public:
  // Default constructor
  Dictionary();
  // Copy constructor
  Dictionary(const Dictionary& org);
  // Destructor
  virtual ~Dictionary();
  // Assignment operator
  virtual Dictionary& operator=(const Dictionary& rhs);
  // Loads words from filename into the dictionary
  virtual bool loadDictionary(const std::string& filename) = 0;
  // Returns true if word is found in the dictionary
  virtual bool find(const std::string& word) const = 0;
};

#endif
```

Implement the push member function for the class given below:

```
template <class T>
class Heap {
public:
  Heap();
  Heap(const Heap<T>& org);
  ~Heap();
  Heap<T>& operator=(const Heap<T>& rhs);
  bool empty() const;
  T& top();
  const T& top() const;
  void pop();
  void push(const T& value);
private:
  std::vector<T> data;
};
```

Implement the destructor for the Set class described in lecture. Recall that the Set<T> and Node<T> classes have the following data members:

```
template <class T>
class Set {
  // ...
private:
  unsigned int sz;
  Node<T>* root;
  Node<T> endNode;
};
```

```
template <class T>
class Node {
public:
  T value;
  Node<T>* right;
  Node<T>* left;
  Node<T>* parent;
  // ...
};
```