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

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

Suppose the default constructor allocates space for 16 elements, sets the capacity to 16 and the numberOfElements to 0. Implement the destructor for this class. Be sure to identify any other assumptions that you make about the class.

Consider the following code with the indicated fix:

```
// ...
// Searches each line of text.
do
{
  // Gets a line from the file.
  notEndOfInFile = inFile.getline(currentLine);

  // Initializes a boolean variable.
  bool foundTarget = false;

  // Keeps track of the element in the target word.
  unsigned int k = 0;

  // Keeps track of the element in the current line.
  unsigned int j = 0;

  // Searches through a single line.
  while(j < currentLine.size() && foundTarget == false)
  {
    // If two characters are not the same, then the placeholder in
    //  the target string is moved back to the first letter.
    if(currentLine[j] != targets[i][k])
    {
      k = 0;
      // This 'if' statement will fix the problem
// FIX HERE
      if(currentLine[j]==currentLine[j-1] && currentLine[j]==targets[i][k])
// FIX HERE
        j--;
// FIX HERE
    }

    // If two letters are the same, then move on to the next
    //  letters.
    else if(currentLine[j] == targets[i][k])
          ++k;

    // If the placeholder reaches the end of the target string,
    //  then the target was found.
    if(k == targets[i].size())
      foundTarget = true;

    // Moves on to the next letter in the line.
    ++j;
  }

  // If the target was found, then the line number is stored.
  if(foundTarget == true)
    results.push_back(lineNumber);

  // Move on to the next line.
  ++lineNumber;
}
// Continues searching text until the end of file is reached
while (notEndOfInFile == true);
// ...
```

Questions on back…

(5 points) Indicate whether or not this FIX solves the problem we discovered in class. If it does fix it, describe the original problem and explain how this fixes it. If it does not fix it, provide an example where it fails and describe how it will fail.

(5 points) Give the big-oh notation time complexity for the above algorithm. Be sure to explain what you are using to describe the size of the input (i.e., what is n?). Justify your answer for full credit.

Give the big-oh notation for the following function.

```
// Sorts large to small
void bubbleSort (vector<double>& vec)
{
  for (int i = vec.size() - 1; i > 0; i--) {
    // move large values to the top
    for (int j = 0; j < i; j++) {
      if (vec[j] < vec[j+1]) {  // swap if out of order
        double temp = vec[j];
        vec[j] = vec[j + 1];
        vec[j + 1] = temp;
      }
    }
  }
}
```

Write a function that accepts a std::string and a std::set<std::string> and returns a boolean value indicating whether or not the string is in the set.

Given the following hash function. Show where the following items would reside when entered into a hash table with 15 buckets.

```
unsigned int hash(const std::string& word)
{
  assert(word.size());
  return (word[0]-'a')*word.size();
}
```

```
the monkey eats pizza like it was a fruit
```

Recall the PQ class developed in lecture yesterday:

```
template <class T>
class PQ
{
public:
  PQ();
  PQ(const PQ<T>& org);
  ~PQ();
  PQ<T>& operator=(const PQ<T>& rhs);
  T top() const;
  void push(const T& val);
  void pop();
private:
  // returns index of the parent of the node corresponding to the
  //  index passed in (or 0 if at the root node)
  unsigned int parent(unsigned int index) const;
  // returns index of the left child of the node corresponding to the
  //  index passed in (or static_cast<unsigned int>(-1) if node doesn't
  //  exist)
  unsigned int lkid(unsigned int index) const;
  // same as lkid but for the right child
  unsigned int rkid(unsigned int index) const;
  // recursively rebalances node and parent node all the way to the
  //  top of the tree
  void rebalanceP(unsigned int index);
  // recursively rebalances node and children nodes all the way to the
  //  bottom of the tree
  void rebalanceKids(unsigned int index);

  std::vector<T> data;
};
```

Consider the following implementation of the pop member function.

```
template <class T>
void PQ<T>::pop()
{
  assert(data.size());
  data[0] = data[data.size()-1];
  data.pop_back();
  rebalanceKids(0);
}
```

Write the rebalanceKids member function.

Put your answer here:

Suppose that a Node class exists with the following data members:

```
Node* parent;
Node* lkid;
Node* rkid;
double value;
```

Suppose that the following are data members of a Tree class:

```
Node* parent;
unsigned int size;
```

and that the following is a member function of the class:

```
void Tree::displaySorted(std::ostream& os)
{
  if(NULL!=parent) {
    displaySortedRecursive(parent, os);
  }
}
```

Write the displaySortedRecursive member function.

Describe (in English) your implementation of the loadDictionary member function for the sorted vector implementation of the dictionary. Give the big-oh time complexity for the function. Be sure to explain your reasoning and list any assumptions you made.