Ane aplpe az deay keps hte docotr awa.
Roemo Tnago Deltta, Thiks iws ase. od yo cpoy???
Hes ist ag grgeat guuy ubt ermj... hte is knid olf stpid
Ii arm sos sawg, lol!! yool! tty ro tnm.
lng
en

alpha
Bravo
bravo
charlie
delta
echo
foxtrot
golf
hotel
india
juliett
kilo
lima
mike
november
oscar
papa
quebec
romeo
sierra
tango
uniform
victor
whiskey
x-ray
yankee
zulu
An
apple
a
day
keeps
the
doctor
away
This
is
base
do
you
copy
He
is
a
great
guy
but
erm
he
is
kind
of
stupid

I
am
so
or
ben
end
ne
long
lang
ling
leng
lung

swag
yolo
lol
dap
ttyl
tmr
tn

```java
/**
 * EECS233 Written HW3
 * Tung Ho Lin
 */

public class PriorityQueue<T extends Comparable<T>> {

  private T[] items;

  private int numItems;

  private int maxItems;

  public PriorityQueue(int maxSize) {
    items = (T[]) new Comparable[maxSize];
    maxItems = maxSize;
    numItems = 0;
  }

  private boolean isEmpty() {
    return numItems==0;
  }

  public void insert(T item) {
    if(numItems == maxItems) {
      T[] olditems = items;
      items = (T[]) new Comparable[numItems*2 + 1];
      for(int i=0; i<olditems.length; i++)
        items[i] = olditems[i];
    }
    items[numItems] = item;
    numItems++;
    siftUp(numItems-1);
  }

  public T removeMax() {
    T toRemove = items[0];
    items[0] = items[numItems-1];
    numItems--;
    siftDown(0);
    return toRemove;
  }

  public void siftUp(int i) {
    T toSift = items[i];
    int child = i;
    int parent = i/2;
    while(parent > 0 && items[child].compareTo(items[parent]) > 0) { //if the child is larger than the parent
      items[child] = items[parent];
      items[parent] = toSift;
      child = parent;
      parent = child/2;
    }
    items[parent] = toSift;
  }

  public void siftDown(int i) {
    T toSift = items[i];
    int parent = i;
    int child = parent*2 + 1;  //child to compare with; start with left child
    while(child < numItems) {
      if(child + 1 < numItems && items[child].compareTo(items[child + 1] < 0)) //if the right child exists and is larger than the left child
        child += 1;
      if(toSift.compareTo(items[child]) >= 0) //if the parent is larger or equal to the child
        break;  //siftDown is complete
      items[parent] = items[child];
      items[child] = toSift;
      parent = child;
      child = parent*2 + 1;
    }
      items[parent] = toSift;
  }
}
```

```java
/**
 * EECS233 WrittenHW3
 * Tung Ho Lin
 * Spelling Checker: Dictionary
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

//using a chaining/bucket hashtable to implement the words database
//the hastable will contain 26 slots: initials a-z
/**
 * The Dictionary text file has to be written in: one word one line format
 * for the build function to work
 */
public class Dictionary {

  private MyBucket[] alphabets;  //a chaining hastable, each slot points to a bucket

  public Dictionary() {
    alphabets = new MyBucket[26]; //initials a-z
    for(int i=0; i<26; i++)
      alphabets[i] = new MyBucket(null);
  }

  public void addWord(String input) {
    String word = input.toLowerCase();
    word = word.replaceAll("\\s*\\p{Punct}+\\s*$", "");  //remove all whitespaces and punctuation at the end of the word
    int firstchar = word.charAt(0);
    if(firstchar >= 'a' && firstchar <='z')  //if the first char is within range
      alphabets[firstchar - 'a'].add(word);  //find the appropriate slot for the word
    else
      return;
  }

  public boolean findWord(String input) {
    String word = input.toLowerCase();
    int firstchar = word.charAt(0);
    if(firstchar >= 'a' && firstchar <='z') {

      return alphabets[firstchar - 'a'].contains(word);  //find the word in the appropriate slot
    }
    else
      return false;
  }

 //limitation of this method, see top
  public void build(String inputfile) {
    try {
    BufferedReader reader = new BufferedReader(new FileReader(inputfile));
    String word;
    while((word = reader.readLine()) != null)
      addWord(word);
    reader.close();
    }
    catch (IOException e) {
      System.err.println("File not found!");
    }
  }


  //inner class Bucket that contains a bunch of nodes
  public class MyBucket {
    private MyNode top;

    public MyBucket(MyNode top) {
      this.top = top;
    }

    public boolean isEmpty() {
      return top == null;
    }

    public void print() {
      String content = "";
      MyNode current = top;
      while(current != null) {
        content = content + current.data;
        current = current.next;
      }
    }

    public boolean contains(String input) {
      if(isEmpty())
        return false;
      MyNode current = top;
      while(current != null) {
        if(current.data.equals(input))
          return true;
        current = current.next;
      }
      return false;
    }

    public void add(String input) {
      if(contains(input))
        return;
      top = new MyNode(input, top);
    }
  }

    //inner class Node that contains a word
    public class MyNode {

      private String data;

      private MyNode next;

    public MyNode(String data, MyNode next){
      this.data = data;
      this.next = next;
    }
  }
}
```

```java
/**
 * EECS233 WrittenHW3
 * Tung Ho Lin
 * Spelling Checker: SpellChecker
 */
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.ArrayList;

//D:\\School Documents\\EECS233\\HW5\\input2.txt
public class SpellChecker {

  private String inputfile;

  private Dictionary dict;

  public SpellChecker(String main, String personal, String input) {
    this.inputfile = input;
    dict = new Dictionary();
    dict.build(main);
    dict.build(personal);
  }

  //main method
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Please enter the path of the main dictionary file.");
    String main = sc.nextLine();
    System.out.println("Please enter the path of the personal dictionary file.");
    String personal = sc.nextLine();
    System.out.println("Please enter the path of the text file to be spell
checked.");
    String input = sc.nextLine();
    SpellChecker checker = new SpellChecker(main, personal, input);
    checker.spellCheck();
  }

  //method to spell check and print out the errors
  public void spellCheck() {
    try {
    Scanner linesc = new Scanner(new File(inputfile));
    int linenumber = 0;
    String line;
    String word;
    while(linesc.hasNextLine()) {
      line = linesc.nextLine();   //scan a whole line of text
      linenumber++;  //increment line number
      Scanner wordsc = new Scanner(line);
      while(wordsc.hasNext()) {
        word = wordsc.next().toLowerCase();  //scan a word from the line of text
        word = word.replaceAll("\\s*\\p{Punct}+\\s*$", "");
        if(dict.findWord(word) == false) {  //if the word is not found in the built
dictionary
          System.out.println(word + " : in Line " + linenumber + " is not spelled
correctly.");
          System.out.println(appendSuggestions(word) + "\n");
        }
      }
      wordsc.close();
    }
    linesc.close();
    System.out.println("Spellcheck completed!");
    }
    catch (FileNotFoundException e) {
      System.err.println("File not found!");
    }
  }

  //collect all the suggestions from the 3 methods and delete identical suggestions
  public String appendSuggestions(String input) {
    ArrayList<String> add = addChar(input);
    ArrayList<String> remove = removeChar(input);
    ArrayList<String> swap = swapChar(input);
    ArrayList<String> suggestions = new ArrayList<String>();
    String output = "Suggestions: ";
    for(int i=0; i<add.size(); i++)  //add all suggestions from first method
      suggestions.add(add.get(i));
    for(int i=0; i<remove.size(); i++) {  //add non-recurrent suggestions
      if(suggestions.contains(remove.get(i)) == false)
        suggestions.add(remove.get(i));
    }
    for(int i=0; i<swap.size(); i++) {  //add non-recurrent suggestions
      if(suggestions.contains(swap.get(i)) == false)
        suggestions.add(swap.get(i));
    }
    for(int i=0; i<suggestions.size()-1; i++) {  //append a String of suggestions
      output += suggestions.get(i) + ", ";  //do not print out ", " on the last word
    }
    try{
    output += suggestions.get(suggestions.size()-1);  //deal with the last word in the list
    }
    catch(ArrayIndexOutOfBoundsException e){
      System.err.println("Suggestions cannot be generated by the built-in methods, Sorry!");  //if the misspelled word cannot be fixed by the 3 methods
      output += "N/A";
    }
    return output;
  }

  //create suggestions by adding a character to anywhere in the word each time
  public ArrayList<String> addChar(String input) {
    ArrayList<String> suggestions = new ArrayList<String>();
    char[] alphabets = new char[26];
    for(int i=0; i<alphabets.length; i++)  //create an array of all 26 alphabets
      alphabets[i] = (char)('a'+ i);
    for(char c : alphabets) {  //for each alphabet to be inserted in the word
      for(int i=0; i<=input.length(); i++) {  //for each space in the word to be inserted
        String suggest = input.substring(0, i) + c + input.substring(i, input.length()); //insert the character between each adjacent characters
        if(dict.findWord(suggest))  //check if it is a correct word
          suggestions.add(suggest);
      }
    }
    return suggestions;
  }

  //create suggestions by removing a character from anywhere in the word each time
  public ArrayList<String> removeChar(String input) {
    ArrayList<String> suggestions = new ArrayList<String>();
    for(int i=0; i<input.length(); i++) {
      StringBuilder builder = new StringBuilder(input);
      builder.deleteCharAt(i);  //delete one character each time
      String suggest = builder.toString();
      if(dict.findWord(suggest))  //compare to the dictionary if it is a correct word
        suggestions.add(suggest);
    }
    return suggestions;
  }

  //create suggestions by swapping 2 characters in a word
  public ArrayList<String> swapChar(String input) {
    ArrayList<String> suggestions = new ArrayList<String>();
    for(int i=0; i<input.length()-1; i++) {
      char[] decon = input.toCharArray();
      char swap = decon[i];
      decon[i] = decon[i+1];  //swapping the characters
      decon[i+1] = swap;
      String suggest = new String(decon);  //back to String
      if(dict.findWord(suggest))  //check if it is a correct word
        suggestions.add(suggest);
    }
    return suggestions;
  }
}
```