COMP 15
Solutions for Sample Exam Problem

I. Sample run: (user input is shown in bold)

```
Welcome to typing practice.
Letters remaining: abcdefghijklmnopqrstuvwxyz
Type some text:  The quick brown fox
Letters remaining: adgjmpstvyz
Type some text:  jumps over the
Letters remaining: adgyz
Type some text:  lazy dog!
Practice is finished.
```

```
public class Main
{
      public static void main(String[] args) {
            System.out.println("Welcome to typing practice.");
            practice();
            System.out.println("Practice is finished.");
      }

      // accept input text until all characters seen.
      private static void practice() {
            Seen seenSoFar = new Seen();

            boolean done = false;
            while (!done)
            {
                  // print characters remaining
                  seenSoFar.printUnseen();

                  // get a line of text
                  String s = getInput("Type some text:  ");
                  String t = s.toLowerCase();

                  // update each lower case character in input as seen
                  for (int i = 0; i < t.length(); i++)
                  {
                        seenSoFar.update(t.charAt(i));
                  }

                  // are we done now?
                  done = seenSoFar.allSeen();
            }
      }

      // getInput:  print prompt and return text entered as a string.
      private static String getInput(String prompt) {
            // you may assume this method is properly defined
      }
}
```

```
public class Seen
{
	private final char [] alphabet =
		{'a','b','c','d','e','f','g','h','i','j','k','l','m','n',
		 'o','p','q','r','s','t','u','v','w','x','y','z'};
	private boolean [] alreadySeen;

	// constructor:  initially all characters are unseen
	public Seen() {
		alreadySeen = new boolean [alphabet.length];
		for (int i = 0; i < alreadySeen.length; i++) {
			alreadySeen[i] = false;
		}
	}

	// mark char c as seen
	public void update(char c) {
		// find c in alphabet.  If found, change alreadySeen to true
		for (int j = 0; j < alphabet.length; j ++)
		{
			if (alphabet[j] == c)
			{
				alreadySeen[j] = true;
			}
		}
	}

	// displays all characters remaining unseen
	public void printUnseen() {
		System.out.print("Letters remaining:  ");
		for (int j = 0; j < alphabet.length; j++)
		{
			if (!alreadySeen[j])
			{
				System.out.print(alphabet[j]);
			}
		}
		System.out.println();
	}

	// have all characters been seen?
	public boolean allSeen() {
		for (int j = 0; j < alphabet.length; j++)
		{
			if (!alreadySeen[j])
			{
				return false;
			}
		}
		return true;
	}
}
```

II. :

```
type Unseen   = String
type Word     = String
type WordList = [Word]

-- remove u w yields the list of characters in u except for those that appear in w.
remove :: Unseen -> Word -> Unseen
remove u w
  = [ c | c <- u, ! elem c w]


-- score u w counts how many characters in u would be removed by word w.
score :: Unseen -> Word -> Int
  = length [ c | c <- u, elem c w]


-- bestWord u ws is a word in ws with highest score, i.e. a word that contains
-- a maximal number of unseen characters.
bestWord :: Unseen -> WordList -> Word
bestword u ws
  head [ w | (w,s) <- zip ws scores, s == maxscore]
  where
     scores   = [ score u w | w <- ws]
     maxscore = maximum scores
```
