APCS 2016 Solutions (Litvin)

A1 Part (a)

```
public class RandomStringChooser
{
  private ArrayList<String> words;

  public RandomStringChooser(String[] wordArray)
  {
    words = new ArrayList<String>();
    for (String w : wordArray)
      words.add(w);
  }

  public String getNext()
  {
    if (words.size() == 0)
      return "NONE"; [1]
    int i = (int)(Math.random() * words.size());
    return words.remove(i); [2]
  }
} [3]
```

Notes:

1. Must return "NONE" if the list is empty.
2. Recall that `remove(i)` returns the element formerly at index `i`.
3. Since the class implements a list, we could also <u>derive</u> this class from `ArrayList<String>`:

   ```
   public class RandomStringChooser extends ArrayList<String>
   {
     public RandomStringChooser(String[] wordArray)
     {
       for (String w : wordArray)
         add(w);
     }

     public String getNext()
     {
       if (size() == 0)
         return "NONE";
       int i = (int)(Math.random() * size());
       return remove(i);
     }
   }
   ```

AI Part (b)

```
  public RandomLetterChooser(String str)
  {
    super(getSingleLetters(str)); [1]
  }
```

Notes:

1. We have to somehow pass the array of letters to `RandomStringChooser`'s constructor, and `super` must be the first statement in the subclass's constructor.

A2 Part (a)

```
public LogMessage(String message)
{
  int i = message.indexOf(":");
  machineId = message.substring(0, i);
  description = message.substring(i+1);
}
```

A2 Part (b)

```
public boolean containsWord(String keyword)
{
  return (" " + description + " ").indexOf(" " + keyword + " ") >= 0;
} 1
```

Notes:

1. It is much easier to pad `description` with spaces at each end than to consider special cases when `keyword` is at the beginning or at the end of `description`. The brute-force alternative is time-consuming and prone to errors:

```
public boolean containsWord(String keyword)
{
  int len = keyword.length();
  String d = description;
  while (true)
  {
    int i = d.indexOf(keyword);
    if (i < 0)
       return false;
    if ((i == 0 || d.substring(i-1, i).equals(" ")) &&
        (i == d.length() - len || d.substring(i + len, i + len + 1).equals(" ")))
        return true;

    /* Or, outside of the AP subset:
    if ((i == 0 || d.charAt(i-1) == ' ') &&
        (i == d.length() - len || d.charAt(i + len) == ' '))
          return true; */

    d = d.substring(i + len);
  }
}
```

A2 Part (c)

```
public List<LogMessage> removeMessages(String keyword)
{
  List<LogMessage> removed = new ArrayList<LogMessage>();
  int i = 0;
  while(i < messageList.size())
  {
    LogMessage msg = messageList.get(i);
    if (msg.containsWord(keyword))
    {
      removed.add(msg);
      messageList.remove(i);
    }
    else
      i++;
  }
  return removed;
} 1
```

Notes:

1. You might be tempted to traverse `messageList` in reverse, but then you need to insert removed elements at the beginning of the `removed` list, which is inefficient.

A3 Part (a)

```
private boolean toBeLabeled(int r, int c, boolean[][] blackSquares)
{
  return !blackSquares[r][c] &&
      (r == 0 || blackSquares[r-1][c] || c == 0 || blackSquares[r][c-1]);
}
```

A3 Part (b)

```
public Crossword(boolean[][] blackSquares)
{
  int rows = blackSquares.length;
  int cols = blackSquares[0].length;

  puzzle = new Square[rows][cols];
  int num = 1;

  for (int r = 0; r < rows; r++)
  {
    for (int c = 0; c < cols; c++)
    {
      if (toBeLabeled(r, c, blackSquares))
      {
        puzzle[r][c] = new Square(false, num);
        num++;
      }
      else
        puzzle[r][c] = new Square(blackSquares[r][c], 0);
    }
  }
}
```

A4 Part (a)

```
public static int totalLetters(List wordList)
{
  int count = 0;

  for (String word : wordList)
    count += word.length();

  return count;
}
```

A4 Part (b)

```
public static int basicGapWidth(List wordList,
                                int formattedLen)
{
  return (formattedLen - totalLetters(wordList)) / (wordList.size() - 1); [1]
}
```

Notes:

1. The number of gaps is one less than the number of words

A4 Part (c)

```
public static String format(List<String> wordList, int formattedLen)
{
  int gapWidth = basicGapWidth(wordList, formattedLen); 1
  String gap = "";
  for (int count = 0; count < gapWidth; count++)
    gap += " ";

  int extraSpaces = leftoverSpaces(wordList, formattedLen);

  String formattedStr = "";

  for (int i = 0; i < wordList.size() - 1; i++)
  {
    formattedStr += wordList.get(i) + gap;
    if (extraSpaces > 0) 2
    {
      formattedStr += " ";
      extraSpaces--;
```