```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xpath-default-namespace="http://earth.google.com/kml/2.2"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math" exclude-result-prefixes="xs math"
    xmlns="http://earth.google.com/kml/2.2"
    version="3.0">   <!--ebb: This is an identity-transformation stylesheet designed to
transform KML into KML, and to draw from a second KML file to import information.-->
<!--ebb: Note that we need to add an XPath default namespace for KML (for input) and an xmlns for the output KML,
too. All files we're processing need to be in KML format: note the two separate declarations of KML in the
<xsl:stylesheet> element. (The output method has to be xml because there's no recognized "kml" value for the @method
on xsl:output.) -->

    <xsl:output method="xml" indent="yes"/>

    <xsl:variable name="PO" select="document('Placeography.kml')" as="document-node()"/>
    <!--ebb: This variable actually invokes a second document that we're going to pull into this transformation! We will be
able to process and mingle information from more than one file by establishing variables like this, set at the document
node (or root element) of the file. We'll invoke it below using $PO. -->
    <xsl:template match="node() | @*">
        <xsl:copy>
            <xsl:apply-templates select="node() | @*"/>
        </xsl:copy>
    </xsl:template>

   <xsl:template match="description">
      <name><xsl:value-of select="."/></name>
   </xsl:template>
    <!--ebb: In the initial output from GPS Visualizer, we discover that the contents of the <description> element contains
a little more detail than the original value we input for <name>. So we write a template rule to match on <description>
and output it as <name>, and we'll suppress the original <name> element from appearing in the output with the next
template rule. -->

  <xsl:template match="Placemark/name">
     <description>
         <xsl:value-of select="$PO//name[. = current()]/following-sibling::description"/>
     </description>
    </xsl:template>

   <!--ebb: Initially we wrote this:
<xsl:template match="name"></xsl:template>
That line is basically an empty template match, so it matches on name and makes it so nothing comes out.
But we decided we wanted to add a new <description> element, pulling from Cassie and Ryan's XML markup, to collect
the text immediately surrounding the location name. So with this template rule we're invoking a second file, signalled by
the variable $PO which we defined at the top of our XSLT document. Note that we are establishing a match-up here
between the two files, so we need to use both dot (.) and current() to be very precise about our match-up: The dot (.) refers
to *wherever you are at the moment in your current path* so it's in-the-moment, here, referring to the most immediate
reference point: the $PO//name. We need current() to signal where we are with the template match, the current context
node of the template match in our main import file from the GPS Visualizer. -->
</xsl:stylesheet>
```