Part 2a. Finding Genes From Coordinates

Your program for project part 2 should handle all of the commands that are described in part 1, plus the following. Commonly, a biologist is interested in knowing all the genes that overlap a given range. In this part of the project, you should implement an interval tree and a 1-dimensional range tree to help respond to the following commands:

- **GenesCrossing x**: print a list of genes that cover position $x$. This is a stabbing query. The genes should be output in the same format as was used in the PrintGeneInfo command from part 1. For example, given these intervals:

  
  \[
  \begin{matrix}
  \text{yaaX, b0005} & \text{talB} \\
  1 & 2 & 3 & 4 & 5 & \ldots
  \end{matrix}
  \quad
  \begin{matrix}
  \text{thrL, b0001} & \text{carB} \\
  15
  \end{matrix}
  \quad
  \begin{matrix}
  \text{dnaI, b0015, Hsp40}
  \end{matrix}
  \]

  The query GenesCrossing 15 should print:

  10 16 thrL b0001
  13 15 nhaA
  14 20 carB

- **GenesIntersectingRange x1 x2**: print a list of genes that have any part of their interval inside the given range. If $x1$ or $x2$ coincides with one of the endpoints of a gene, that gene should be output. Again, the format for listing the genes should be the same as used in PrintGeneInfo. For example, the output of the range query:

  \[
  \begin{matrix}
  \text{yaaX, b0005} & \text{talB} \\
  1 & 2 & 3 & 4 & 5 & \ldots
  \end{matrix}
  \quad
  \begin{matrix}
  \text{thrL, b0001} & \text{carB} \\
  15
  \end{matrix}
  \quad
  \begin{matrix}
  \text{dnaI, b0015, Hsp40}
  \end{matrix}
  \]

  should be:

  3 7 yaaX b0005
  5 6 talB

- **PrintIntervalTree**: print a representation of the current interval tree. To do this, perform an inorder traversal. For each node, print the value of $x_{\text{med}}$ used to divide the intervals, as well as a list of the intervals (genes) that are split at that node. For example, assuming the genes shown in the figures above, if the current node splits on position 15, you should print:

  15 thrL nhaA carB

  Again, if an interval has more than one name, pick an arbitrary one to print.

You should modify your code that handles AddGene and RemoveAlias so that it invalidates your interval and range trees if a gene is added or deleted. If the interval or range trees are invalid when one of the commands in Part 2a is issued, then you should rebuild the tree. **You do not need to support dynamic insert or delete within your interval or range trees for this part.** Just rebuild the trees as needed (but not more often than that). Your 1-d range tree should be a balanced tree (such as a splay tree).
Part 2b. Genome rearrangement queries

Almost done!

Over the course of evolution, genes can move around along the genome. Given two genomes from similar species, a biologist would like to ask, for example, “which genes are at the end of the first genome but the beginning of the second?” This can be translated into a 2-dimensional range query as shown below, where a dot at \((x, y)\) represents a gene that is at position \(x\) in genome A but position \(y\) in genome B.

You will implement a 2-dimensional \textit{kd-tree} to store points \((x, y)\) where \(x\) represents the location of a gene in one genome while \(y\) represents the location of the corresponding gene in the second genome. You will then have to respond to rectangular range queries.

You should handle the following commands:

- Genome A/B: Up to this point, we assumed all commands applied to the same genome. Now we need to know which genome a command applies to. After issuing the command genome A all commands should be applied to the first genome, while after the command genome B all commands should be applied to the second genome.

- PairGenes name1 name2: add a point \((x, y)\) to your 2-d range tree, where \(x\) is the floor of the midpoint of the interval for the gene named name1 in genome A, and \(y\) is the floor of the midpoint of the interval for the gene named name2 in genome B. If either name1 or name2 are not known genes, report an error.

- UnpairGenes name1 name2: remove the point that corresponds to the two genes name1 and name2. This is the inverse of the PairGenes command.

- GenesInRectangle \(x_1 \ x_2 \ y_1 \ y_2\): print the gene information for each of the points that fall in the rectangle \([x_1, x_2] \times [y_1, y_2]\). When printing a gene that lies within the given rectangle, you should print two lines for each gene:

\[
\begin{align*}
A \ & x_1 \ x_2 \ nameA1 \ nameA2 \ nameA3 \ldots \\
B \ & y_1 \ y_2 \ nameB1 \ nameB2 \ nameB3 \ldots
\end{align*}
\]

where A and B are literal strings, \([x_1, x_2]\) is the range of the gene in genome A, \([y_1, y_2]\) is the range of the gene in genome B, and nameA1, nameA2, ... and nameB1, nameB2, ... are the names of the gene in each genome.

For this part, you may assume that the command RemoveAlias is never called after the command PairGenes has been issued for the first time.