Functional Decomposition

Mikayla Timm

Project 1

**HASH**: Adapted from DSA1 Project.

* HashP newHash(int)
* int findMatchesHash(HashP, char \*, int)
* void insertHash(HashP, char \*, int)
* void removeHash(HashP, char \*)
* void displayHash(HashP)
* void displayStatisticsHash(HashP)
* HashP freeHash(HashP)

New Functions:

* HashP insertPhrases(HashP, char \*\*, int, int, int)
* int getSimilarity(HashP, char \*\*, int, int, int)

**NODE**: Adapted from DSA1 Project.

* NodeP getLeft(NodeP)
* void setLeft(NodeP, NodeP)
* NodeP getRight(NodeP)
* void setRight(NodeP, NodeP)
* char \*getPhrase(NodeP)
* void displayNodeInfo(NodeP)
* NodeP freeNode(NodeP)
* int comparePhrases(char \*, char \*)

**FILEIO**:

* int setupInputFile()
* FILE \*openInputFile()
* char \*\*getFileNames(FILE \*inputfile, int numFiles, int MAXNAMELEN)
* char \*getAllChars(char \*masterBuffer, char \*filename, int MAXNAMELEN)
* char \*\*freeFileNames(char \*\*fileNames, int numFiles)

**STRINGPROCESSING**:

* char \*initializeMasterBuffer(int MAXCHARSPERFILE)
* WordsP newWords(int MAXCHARSPERFILE, int MAXCHARSPERWORD)
* char \*\*getTokenizedWords(WordsP words)
* int getNumWords(WordsP words)
* WordsP tokenize(WordsP words, char \*allChars, int MAXCHARSPERFILE, int MAXCHARSPERWORD)

**MAIN (ComparingFilesTest)**:

* void printResults(int \*\*, int)
* int \*\*initializeSimilarityMatrix(int)