The English Language
Language

Team Members:
Emily Bau, eb3029, Project Manager
Nivita Arora, na2464, Language Guru
Michele Lin, ml3706, System Architect
Candace Johnson, crj2121, System Architect
Rabia Akhtar, ra2805, Tester

Introduction:

The English Language language solves problems specific to document manipulation and data extrapolation. People who would like to write scripts that analyze multiple documents quickly and can cross compare documents will find it it hard in traditional languages. Our language provides core file manipulation operations and storage structures and allows for libraries that mine statistics and check for plagiarism. This could especially be useful for teaching and publishing related activities.

Language Overview:

Our language allows for streamlining of calculating most used words, most popular subject, time a human takes to read this file, and other useful information related to one text file. It will also help streamline comparing lists of files for searching for relevant keywords and other comparison functions.

Data Types:

integers	int x = 4
floats	float y = 3.14159
string	string = “hello world”
arrays	string [] = [“hello”, “world”]
booleans	bool isCommon = true

Objects:

document	doc = new doc(String path); A document object containing a max-heap and hashmap (word, count) to store and process file content. It also contains information like number of words.
max-heap	max-heap storing (count, word), maximizing by count
hashmap	hashmap storing (word, count)

Operators:

Integer Operators:

==, <, >, <=, >=	Numerical relation
+, -, *, /, %, ++, +=	Arithmetic

General Operators:

=	equals

String Operators:

+	concatenation
==	is equal to
<	shorter in length
>	longer in length
[]	index of string

Logical Operators:

||	or
&&	and
!	not

Control Flow:

/* to */	Multiple line comments
//	Single line comments
if/elif/else	Conditional statements
for/while	Conditional loop statements
;	End of statement

Built in Functions:

print()	Function to print any data type
doc.getKeywords(int numberOfKeywords)	Return the most frequently used words in a document
toLower()	Changes an uppercase word to lowercase
doc.getCount(string word)	Searches for a particular word in a document and returns the frequency of word (integer)

Sample Program:

```
boolean plagarismCheck(doc EssayA, doc EssayB){

	string [] keyWordsA = EssayA.getKeywords(10);
	string [] keyWordsB = EssayA.getKeywords(10);
	int  totalSimilar = 0;

	for(int i =0; i<9; i++){
		for(j=0; j<9; j++){
			if(keyWordsA[i]==keyWords[j]){
				totalSimilar++;
			}			}
	}

	if(totalSimilar >7){
		return True;
	}
	return False;
}

doc findRelevant (doc[] docs, string keyword) {
	doc mostRelevant;
	int keywordCount = 0;
	for (int i = 0; i < docs.length; i++) {
		if (docs[i].getCount(keyword) > keywordCount) {
			mostRelevant = docs[i];
		}
```

```
        }
        return mostRelevant;
}
```