API OF THE DISTINGO INDEX LIBRARY
VERSION 2.0

A. List of constants

1. Constants defining the language used in the relatedness functions. Please note that French is currently not available.

```
#define NSIMEnglish     1
#define NSIMFrench      2
```

2. Other constants used in the relatedness functions

```
#define maxLen          70

#define maxPathLen      30
#define maxConstDist    8


#define horizontalDirection   0
#define downwardDirection     1
#define upwardDirection       2
#define initialDirection      3
```

`maxLen` indicates the maximum size of a word returned by Distingo (path or list of words). It is also the maximum size of a word given as a parameter to Distingo.
`maxPathLen` indicates the maximum length of the path returned by Distingo
`maxConstDist` indicates the maximum semantic distance or relatedness that Distingo can compute between two words.
IMPORTANT NOTE: Even if maxConstDist can be set up to 8, we recommend using 5 as a maximum. Otherwise the result can be slow and require a great deal of memory (for the functions that return the list of words).
`horizontalDirection` indicates the horizontal direction (synonym or antonym) in a path between two words
`downwardDirection` indicates the downward direction (hyponym) in a path between two words
`upwardDirection` indicates the upward direction (hypernym) in a path between two words
`initialDirection` is a pseudo direction in a path between two words, just used for the first word in the path.

B. Types of data

```
typedef struct {
      char word[maxLen];
      short senseIndex;
      long direction;
} onePath;
```

`onePath` describes one word of the path along with the direction used to reach this word.

C. List of functions

Parameters passed to entries are prefixed by « -> », and those returned are prefixed by « <-».
Parameters that are both passed to and returned entries are prefixed by « <-> ».

```
PrefixFunc(short)  SetRegistrationKey(char key[LenMaxKey]);
```

This function allows one to enter the activation key for the Distinguo Index library.
-> `key` contains the activation key
<-return
The return is 1 if the registration is successful, 0 otherwise.

```
PrefixFunc(void) loadNSIMData(short languages);
```

This function initializes the engine of Distingo
-> languages contains the value of the working languages desired (French, English or French+English)

```
PrefixFunc(void) setNSIMLanguage(short language);
```

This function allows one to change the working language.
-> language equals either French or English.

```
PrefixFunc(short) getVerbSensesCount(char *word);
```

This function computes the number of senses of a verb.
-> word contains the verb in the form of a zero-terminated string.
<-return
The return of the function is the number of senses of the verb
The function returns -10 if word is unknown

.

```
PrefixFunc(short) getVerbSensesDefinition(char *word, char
*senseDefinition[],short senseDefinitionSize)
```

This function returns the definition of the senses of a verb.
-> word contains the verb in the form of a zero-terminated string.
<- senseDefinition contains in the return the definitions of the verb
-> senseDefinitionSize contains the size of the senseDefinition array

<-return
The function returns 0 if word is known and the definitions are returned in the senseDefinition array
The function returns -10 if word is unknown

```
PrefixFunc(short) getRelatednessVerb (short constDist, char *word1, short
sense1, char *word2, short sense2);
```

This function computes the relatedness between two verbs. The relatedness value is given by constDist - distance. The maximal relatedness is given by constDist and the minimum is given by zero (no relatedness).

-> constDist contains the maximum relatedness. The maximum value for constDist is maxConstDist.
-> word1 contains the first verb in the form of a zero-terminated string.
-> sense1 contains the sense index of the first verb or -1 for all senses
-> word2 contains the second verb in the form of a zero-terminated string.
-> sense2 contains the sense index of the second verb or -1 for all senses

<- return : the return of the function is the relatedness between the two verbs (word1 and word2).
The function returns -10 if word1 is unknown
The function returns -20 if word2 is unknown
The function returns -30 if there is no database in Distinguo for getting the relatedness between verbs
The function returns -40 if constDist is strictly superior to maxConstDist.

```
PrefixFunc(short) getRelatednessVerbPath (short constDist, char *word1,
short sense1, char *word2, short sense2, onePath path[maxPathLen], short
*pathLen);
```

This function computes the relatedness between two verbs. The relatedness value is given by constDist - distance. The maximal relatedness is given by constDist and the minimum one is given by zero (no relatedness).
This function also returns the path (the list of intermediate words along with the direction to each of them) from word1 to word2. This path is returned in the path parameter and the length of the path is given by pathLen.

-> constDist contains the maximum relatedness. The maximum value for constDist is maxConstDist.
-> word1 contains the first verb in the form of a zero-terminated string.
-> sense1 contains the sense index of the first verb or -1 for all senses
-> word2 contains the second verb in the form of a zero-terminated string.
-> sense2 contains the sense index of the second verb or -1 for all senses
<- path contains an array of onePath of size maxPathLen.
<- pathLen is the size of the path.

<- return : the return of the function is the relatedness between the two verbs (word1 and word2).
The function returns -1 if word1 is unknown
The function returns -2 if word2 is unknown
The function returns -3 if there is no database in Distinguo for getting the relatedness between verbs
The function returns -4 if constDist is strictly superior to maxConstDist.

```
PrefixFunc(short) getRelatednessVerbList (short maxDist, char *word, short
sense, long nbMaxSimWords, long *nbSimWords, char simWords[][maxLen], short
simWordsDist[], short simWordsPos[], short simWordsSense[]);
```

This function returns the list of verbs having a maximum distance of maxDist from the verb contained in the parameter word.

-> maxDist contains the maximum distance. The maximum value for maxDist is maxConstDist.
-> word contains the verb in the form of a zero-terminated string.
-> sense contains the sense index of the verb or -1 for all senses
-> nbMaxSimWords contains the maximum number of elements that simWords and simWordsDist can contain
<- nbSimWords contains the number of verbs having a maximum distance of maxDist
<- simWords is an array containing the verbs having a maximum distance of maxDist
<- simWordsDist is an array containing, for each verb in simWords (with the same index), its exact distance from word.

<- return : the return of the function is zero if there is no error.
The function returns -1 if word is unknown
The function returns -2 if the number of verbs having a maximum distance of maxDist is greater than nbMaxSimWords
The function returns -3 if there is no database in Distingo for getting the relatedness between verbs
The function returns -4 if maxDist is strictly superior to maxConstDist.

Note : it is possible to call getRelatednessVerbList by setting nbMaxSimWords to 0 and simWords and simWordsDist to NULL. This allows one to get in nbSimWords the exact number of the number of verbs having a maximum distance of maxDist.

```
PrefixFunc(short) getNounSensesCount(char *word);
```

This function is the same as getVerbSensesCount but works with nouns instead.

```
PrefixFunc(short) getNounSensesDefinition(char *word, char
*senseDefinition[],short senseDefinitionSize)
```

This function is the same as getVerbSensesDefinition but works with nouns instead.

```
PrefixFunc(short) getRelatednessNoun(short constDist, char *word1, char
*word2);
```

This function is the same as getRelatednessVerb but works with nouns instead.

```
PrefixFunc(short) getRelatednessNounPath(short constDist, char *word1, char
*word2, onePath path[maxPathLen], short *pathLen);
```

This function is the same as `getRelatednessVerbPath` but works with nouns instead.

```
PrefixFunc(short) getRelatednessNounList(short maxDist, char *word, long
nbMaxSimWords, long *nbSimWords, char simWords[][maxLen], short
simWordsDist[]);
```

This function is the same as `getRelatednessVerbList` but works with nouns instead.

```
PrefixFunc(short) getAdjSensesCount(char *word);
```

This function is the same as `getVerbSensesCount` but works with adjectives instead.

```
PrefixFunc(short) getAdjSensesDefinition(char *word, char
*senseDefinition[],short senseDefinitionSize)
```

This function is the same as `getVerbSensesDefinition` but works with adjectives instead.

```
PrefixFunc(short) getRelatednessAdj(short constDist, char *word1, char
*word2);
```

This function is the same as `getRelatednessVerb` but works with adjectives instead.

```
PrefixFunc(short) getRelatednessAdjPath(short constDist, char *word1, char
*word2, onePath path[maxPathLen], short *pathLen);
```

This function is the same as `getRelatednessVerbPath` but works with adjectives instead.

```
PrefixFunc(short) getRelatednessAdjList(short maxDist, char *word, long
nbMaxSimWords, long *nbSimWords, char simWords[][maxLen], short
simWordsDist[]);
```

This function is the same as `getRelatednessVerbList` but works with adjectives instead.