//makeProper()

    /\*

     Remove:

     duplicates with the lower separation

     word that is empty

     word that contains non-alpha characters

     space

     when word1[a]=word2[b] and word2[a]=word1[b]

     \*/

    const int nPatterns1 = 8;

    char word1[][MAX\_WORD\_LENGTH+1] = {"UCLA","usc","","sunset","foot??ball","Bruins","NEGATIVE"," "};

    char word2[][MAX\_WORD\_LENGTH+1] = {"USC","uClA","empty","sunrise","basketball","trojan","VALUE","space"};

    int sepa1[]={3,1,2,4,3,2,-1,3};

    assert(makeProper(word1, word2, sepa1, nPatterns1)==3);

    assert(strcmp(word1[0],"usc")==0 && strcmp(word2[0],"ucla")==0);

    assert(strcmp(word1[2],"bruins")==0 && strcmp(word2[2],"trojan")==0);

    assert(sepa1[0]==1);

    //rate()

    const int TEST1\_NRULES = 4;

    char test1w1[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] = {

        "mad",       "deranged", "nefarious", "have"

    };

    char test1w2[TEST1\_NRULES][MAX\_WORD\_LENGTH+1] = {

        "scientist", "robot",    "plot",      "mad"

    };

    int test1dist[TEST1\_NRULES] = {

        1,           3,          0,           12

    };

    /\*

     Error Handle:

     Empty document

     size less than or equal to 0

     \*/

    assert(rate("",test1w1,test1w2, test1dist, TEST1\_NRULES) == 0);

    assert(rate("The UCLA scientist Smallberg is MAD!",test1w1,test1w2, new const int(0), TEST1\_NRULES) == 0);

    assert(rate("The UCLA scientist Smallberg is MAD!",test1w1,test1w2, new const int(-1), TEST1\_NRULES) == 0);

    /\*

     Cases:

     document contains a serie of continuous space

     Capital letters

     word pairs showing up for more than once

     similar words - "scientist" and "scientists"

     word pairs separated by punctuations: "deranged-robot"

     \*/

    assert(rate("The mad UCLA scientist unleashed a deranged evil giant robot.",

                test1w1, test1w2, test1dist, TEST1\_NRULES) == 2);

    assert(rate("The mad UCLA scientist unleashed    a deranged robot.",

                test1w1, test1w2, test1dist, TEST1\_NRULES) == 2);

    assert(rate("\*\*\*\* 2018 \*\*\*\*",

                test1w1, test1w2, test1dist, TEST1\_NRULES) == 0);

    assert(rate("  That plot: NEFARIOUS!",

                test1w1, test1w2, test1dist, TEST1\_NRULES) == 1);

    assert(rate("deranged deranged robot deranged robot robot",

                test1w1, test1w2, test1dist, TEST1\_NRULES) == 1);

    assert(rate("That scientist said two mad scientists suffer from deranged-robot fever."

                ,test1w1, test1w2, test1dist, TEST1\_NRULES) == 0);

    assert(rate("mad scientist mad scientists mad scientist"

                ,test1w1, test1w2, test1dist, TEST1\_NRULES) == 1);

    //When there are multiple patterns sharing the same word

    const int TEST2\_NRULES=4;

    char test2w1[][MAX\_WORD\_LENGTH+1]={"ucla","bruin","bruins","sunset"};

    char test2w2[][MAX\_WORD\_LENGTH+1]={"fight","fight","fight","village"};

    int test2dist[TEST2\_NRULES] = {1,3,3,2};

    assert(rate("UCLA Bruins fight! bruin fight!"

                ,test2w1, test2w2, test2dist, TEST2\_NRULES) == 3);

    /\*

     Cases:

     When word1 and word2 are equal

     If document contains words that exceed limit

     \*/

    char test3w1[][MAX\_WORD\_LENGTH+1]={"same"};

    char test3w2[][MAX\_WORD\_LENGTH+1]={"same"};

    int test3dist[TEST2\_NRULES] = {1};

    assert(rate("Same same SAME SaMe",test3w1, test3w2,test3dist,1)==1);

    assert(rate("same ThisStringIsTooLongThatItExceedsLimit same",test3w1, test3w2,test3dist,1)==1);

    cerr << "All tests succeeded" << endl;