/\* Assignment: C5 Post Activity, Task 2

\* File: C5\_PA\_Task2\_katherto.c

\* Date: 17 February 2016

\* By: Kathryn Atherton

\* katherto

\* Section: 03, 1:30-3:20

\* Team: 45

\*

\* ELECTRONIC SIGNATURE

\* KATHRYN ATHERTON

\*

\* The electronic signature above indicates that the

\* program submitted for evaluation is my individual work

\* and I have a general understanding of all aspects of

\* its development and execution.

\*

\* A BRIEF DESCRIPTION OF WHAT THE PROGRAM OR FUNCTION

\* DOES

\* This is a program to take in a keyword, determine the

\* number of non-repeating characters in the keyword, and

\* then assign numberical values to each of the letters

\* based on their position in the alphabet. Determine how

\* many letters are in each group of characters. The user

\* then inputs long string of no more than 1000 characters.

\* The program then outputs the deciphered message.

\*/

#include <stdio.h>

#include <string.h>

#include <math.h>

//FUNCTION PROTOTYPES

double repeating(char \*keyword);

int groupSize(char \*keyword, double nonRepChar);

int ErrorCheck(char \*message, double nonRepChar, int gSize);

//MAIN FUNCTION

int main(void){

//DECLARE VARIABLES

char keyword[21]; //entered by user

char keywordNoRep[21]; //keyword modified to remove the repeating characters

char message[1001]; //message to be decoded, entered by user

char tempLetter;

double nonRepChar;

int compare;

int groups;

int gSize;

int i;

int j;

int k;

int kLength;

int l;

int mLength;

int newKeywordLength;

//USER INPUT PROMPT -- KEYWORD

printf("Enter a keyword: ");

scanf("%s", keyword);

//FUNCTION CALLS --KEYWORD

nonRepChar = repeating(keyword);

gSize = groupSize(keyword, nonRepChar);

//USER INPUT PROMPT -- MESSAGE

printf("Input the message to be decoded: ");

scanf("%s", &message[0]);

//ERROR CHECKING

mLength = ErrorCheck(message, nonRepChar, gSize);

//DECODE MESSAGE

//make array of arrays

//declare array of arrays with sizes based on previous calculations

groups = nonRepChar;

char decode[groups][gSize];

i = 0;

j = 0;

l = 0;

while(i < mLength){

k = 0;

j = 0;

while(k < gSize){

decode[l][j] = message[i];

i++;

k++;

j++;

}

if(l < groups - 1){

l++;

}

else{

break;

}

}

//find length of keyword

i = 0;

kLength = 0;

while(keyword[i] != '\0'){

kLength = kLength + 1;

i++;

}

//take out repeating characters in keyword

j = 0;

i = 1;

newKeywordLength = 0;

char kLetter;

keywordNoRep[0] = keyword[0];

while (i <= kLength){

kLetter = keyword[i];

while (j <= newKeywordLength){

compare = keywordNoRep[j];

if (kLetter == compare){

j = 0;

break;

}

else if (j == newKeywordLength){

keywordNoRep[newKeywordLength + 1] = keyword[i];

newKeywordLength++;

j = 0;

break;

}

else{

j++;

}

}

i++;

}

i = 0;

char temp[gSize];

//swap arrays into alphabetical order of keyword

for(i = 0; i <= newKeywordLength - 1; i++){

j = i + 1;

for (j = i + 1; j <= newKeywordLength; j++){

if(keywordNoRep[i] > keywordNoRep[j]){

tempLetter = keywordNoRep[j];

keywordNoRep[j] = keywordNoRep[i];

keywordNoRep[i] = tempLetter;

k = 0;

while(k < gSize){

temp[k] = decode[i][k];

k++;

}

k = 0;

while(k < gSize){

decode[i][k] = decode[j][k];

k++;

}

k = 0;

while(k < gSize){

decode[j][k] = temp[k];

k++;

}

}

}

}

//OUTPUT STATEMENT

i = 0;

j = 0;

printf("Decoded Message: ");

while(i <= groups){

while(j < gSize){

printf("%c", decode[i][j]);

j++;

}

j = 0;

printf(" ");

i++;

}

printf("\n");

//RETURN STATEMENT

return 0;

}

double repeating(char \*keyword){

//DECLARE VARIABLES

double nonRepChar;

int i;

char letter;

double freqLetter;

double length;

int j;

//INITIALIZE VARIABLES

i = 0;

j = 1;

length = 0.0;

//FIND LENGTH OF KEYWORD

while(keyword[i] != '\0'){

length = length + 1.0;

i++;

}

//FIND NUMBER OF REPEATING CHARACTERS

nonRepChar = length;

freqLetter = 0.0;

for(letter = 'A'; letter <= 'Z'; letter++){

for(i = 0; i <= length; i++){

if(keyword[i] == letter){

freqLetter = freqLetter + 1.0;

}

}

if (freqLetter > 1.0){

nonRepChar -= (freqLetter - 1.0);

}

freqLetter = 0.0;

}

//RETURN STATEMENT

return nonRepChar;

}

int groupSize(char \*keyword, double nonRepChar){

//DEFINE VARIABLES

double sum = 0.0;

int groupSize;

int i = 0;

//FIND SUM

while(keyword[i] != '\0'){

if(keyword[i] == 'A'){

sum = sum + 1.0;

i++;

}

else if(keyword[i] == 'B'){

sum = sum + 2.0;

i++;

}

else if(keyword[i] == 'C'){

sum = sum + 3.0;

i++;

}

else if(keyword[i] == 'D'){

sum = sum + 4.0;

i++;

}

else if(keyword[i] == 'E'){

sum = sum + 5.0;

i++;

}

else if(keyword[i] == 'F'){

sum = sum + 6.0;

i++;

}

else if(keyword[i] == 'G'){

sum = sum + 7.0;

i++;

}

else if(keyword[i] == 'H'){

sum = sum + 8.0;

i++;

}

else if(keyword[i] == 'I'){

sum = sum + 9.0;

i++;

}

else if(keyword[i] == 'J'){

sum = sum + 10.0;

i++;

}

else if(keyword[i] == 'K'){

sum = sum + 11.0;

i++;

}

else if(keyword[i] == 'L'){

sum = sum + 12.0;

i++;

}

else if(keyword[i] == 'M'){

sum = sum + 13.0;

i++;

}

else if(keyword[i] == 'N'){

sum = sum + 14.0;

i++;

}

else if(keyword[i] == 'O'){

sum = sum + 15.0;

i++;

}

else if(keyword[i] == 'P'){

sum = sum + 16.0;

i++;

}

else if(keyword[i] == 'Q'){

sum = sum + 17.0;

i++;

}

else if(keyword[i] == 'R'){

sum = sum + 18.0;

i++;

}

else if(keyword[i] == 'S'){

sum = sum + 19.0;

i++;

}

else if(keyword[i] == 'T'){

sum = sum + 20.0;

i++;

}

else if(keyword[i] == 'U'){

sum = sum + 21.0;

i++;

}

else if(keyword[i] == 'V'){

sum = sum + 22.0;

i++;

}

else if(keyword[i] == 'W'){

sum = sum + 23.0;

i++;

}

else if(keyword[i] == 'X'){

sum = sum + 24.0;

i++;

}

else if(keyword[i] == 'Y'){

sum = sum + 25.0;

i++;

}

else{

sum = sum + 26.0;

i++;

}

}

//FIND GROUP SIZE

groupSize = sqrt(sum / nonRepChar);

//RETURN STATEMENT

return groupSize;

}

int ErrorCheck(char \*message, double nonRepChar, int gSize){

//DECLARE AND INITIALIZE VARIABLES

int i = 0;

int mLength = 0;

int groups = nonRepChar;

while(message[i] != '\0'){

mLength = mLength + 1;

i++;

}

if(mLength != (groups \* gSize)){

printf("\nWarning: Message may have been altered by enemy spies.\n");

}

return mLength;

}