TASK 1 Write a program that calculates the Coleman-Liau Index of some text.

//

// LAB 8 TASK 1

// 5/12/20

// Sidi Liang

//

#include <stdio.h>

#include <string.h>

float calculateCLI(char string[]);

int main(){

char text[100] = "I like cats. Cats like me. Miao miao miao. Dogs are bad. Bad dogs bad.";

char text2[500] = "Tomorrow, and tomorrow, and tomorrow, Creeps in this petty pace from day to day, To the last syllable of recorded time; And all our yesterdays have lighted fools The way to dusty death. Out, out, brief candle. Life's but a walking shadow, a poor player That struts and frets his hour upon the stage And then is heard no more. It is a tale Told by an idiot, full of sound and fury Signifying nothing.";

char text3[1000] = "Existing computer programs that measure readability are based largely upon subroutines which estimate number of syllables, usually by counting vowels. The shortcoming in estimating syllables is that it necessitates keypunching the prose into the computer. There is no need to estimate syllables since word length in letters is a better predictor of readability than word length in syllables. Therefore, a new readability formula was computed that has for its predictors letters per hundred words and sentences per hundred words. Both predictors can be counted by an optical scanning device, and thus the formula makes it economically feasible for an organization such as the US Office of Education to calibrate the readability of all textbooks for the public school system.";

printf("CLI is Grade %f\n", calculateCLI(text));

printf("CLI is Grade %f\n", calculateCLI(text2));

printf("CLI is Grade %f\n", calculateCLI(text3));

}

float calculateCLI(char string[]){

int characters = 0, words = 1, sentences = 0;//Because each sentance begin with words and ends with peroid, words count starts from 1

for(int i = 0; string[i] != '\0'; i++){

if(string[i] != ' ' && string[i] != '.'){

characters++; //Calculate characters

}

if(string[i] == ' '){

words++; //Calculate words

}

if(string[i] == '.'){

sentences++; //Calculate sentences

}

}

float cli = 5.89 \* ((float)characters / (float)words) - 29.5 \* ((float)sentences / (float)words) - 15.8;

if(cli < 1) cli = 1; //If the calculated grade is less than 1, set it to 1 instead of returning a lower number.

return cli;

}

CODE FOR TASK 1

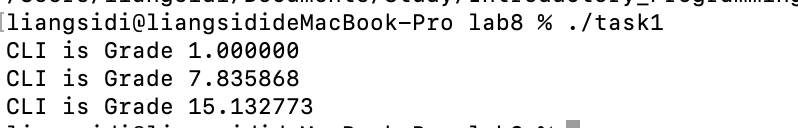


Figure 1: Output of Code in Task 1.

TASK 1.2 Write a program that calculates the amount of coins to return.

//

// LAB 8 TASK 2

// 5/12/20

// Sidi Liang

//

#include <stdio.h>

struct Change {

int paid, itemCost, changeIn200Pence, changeIn100Pence, changeIn50Pence, changeIn20Pence, changeIn10Pence, changeIn5Pence, changeIn2Pence, changeIn1Pence;

};

struct Change getChange(int itemCost, int paid);

void printChange(struct Change change);

int main() {

struct Change coins;

coins = getChange(7, 10);

printChange(coins);

coins = getChange(56, 70);

printChange(coins);

coins = getChange(124, 200);

printChange(coins);

coins = getChange(1232, 2000);

printChange(coins);

getchar();

return 0;

}

struct Change getChange(int itemCost, int paid){

struct Change changeCoins;

int \*changePt = &changeCoins;

for(int i = 0; i<sizeof(struct Change)/sizeof(int); i++)

{

(\*changePt++) = 0;//Initialize every member in struct change

}

changeCoins.paid = paid;

changeCoins.itemCost = itemCost;

int totalChangeInPence = paid - itemCost;

//Calculate the coins from the largest coin to the smallest

while(totalChangeInPence >= 200){

changeCoins.changeIn200Pence += 1;

totalChangeInPence -= 200;

}

while(totalChangeInPence >= 100){

changeCoins.changeIn100Pence += 1;

totalChangeInPence -= 100;

}

while(totalChangeInPence >= 50){

changeCoins.changeIn50Pence += 1;

totalChangeInPence -= 50;

}

while(totalChangeInPence >= 20){

changeCoins.changeIn20Pence += 1;

totalChangeInPence -= 20;

}

while(totalChangeInPence >= 10){

changeCoins.changeIn10Pence += 1;

totalChangeInPence -= 10;

}

while(totalChangeInPence >= 5){

changeCoins.changeIn5Pence += 1;

totalChangeInPence -= 5;

}

while (totalChangeInPence >= 2) {

changeCoins.changeIn2Pence += 1;

totalChangeInPence -= 2;

}

while (totalChangeInPence >= 1) {

changeCoins.changeIn1Pence += 1;

totalChangeInPence -= 1;

}

return changeCoins;

}

void printChange(struct Change change){

printf("Customer gave %d pence, item(s) cost %d pence.\nGive customer:\n", change.paid, change.itemCost);

printf("£2 £1 50 20 10 5 2 1\n%d %d %d %d %d %d %d %d\n", change.changeIn200Pence, change.changeIn100Pence, change.changeIn50Pence, change.changeIn20Pence, change.changeIn10Pence, change.changeIn5Pence, change.changeIn2Pence, change.changeIn1Pence);

puts("");//Change line

}

CODE FOR TASK 2

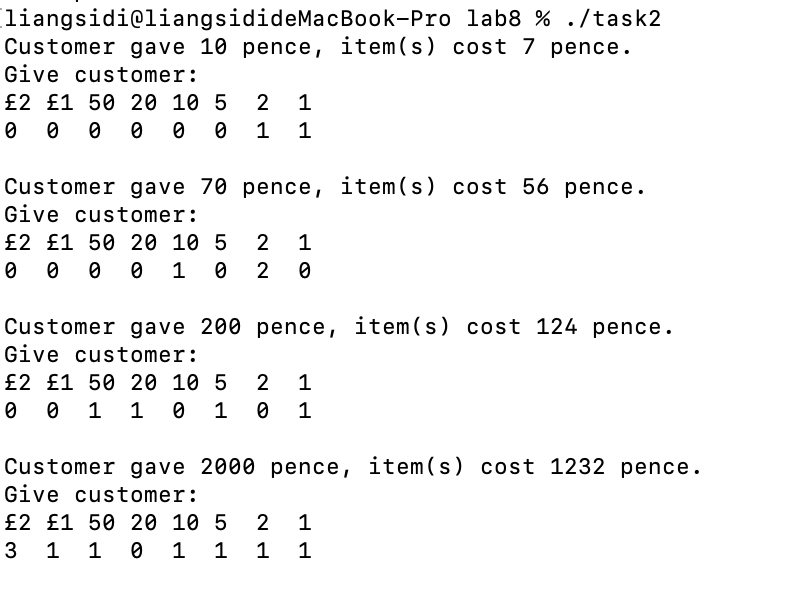


Figure 2: Output of Code in Task 2.

* TASK 3 Write a program that decrypts them.

//

// LAB 8 TASK 2

// 5/12/20

// Sidi Liang

//

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct TwoLetters{

char firstLetter;

char secondLetter;

};

struct TwoLetters split(int decrypted){

struct TwoLetters a;

a.firstLetter = decrypted >> 8;

a.secondLetter = decrypted & 0xFF;

return a;

}

int main() {

char encrypted[1000];

scanf("%[^\n]", encrypted);

int wordCount = 1;

for(int i = 0; encrypted[i] != '\0'; i++){

if(encrypted[i] == ' '){

wordCount++;

}

}

//Seperate the char and convert them to int

char \*token, \*tmpPtr;

token = strtok(encrypted, " ");

long allInts[wordCount];

allInts[0] = strtoul(token, &tmpPtr, 10);

int i = 1;

while(token != NULL) {

//printf( "%s\n", token );

token = strtok(NULL, " ");

if(token != NULL){

allInts[i] = strtoul(token, &tmpPtr, 10);

i++;

}

}

for(int i = 0; i < wordCount; i++){

allInts[i] ^= 31337;

//printf("%ld \n", allInts[i]);

struct TwoLetters result = split((int)allInts[i]);

printf("%c%c", result.firstLetter, result.secondLetter);

}

return 0;

}

Code for TASK 3

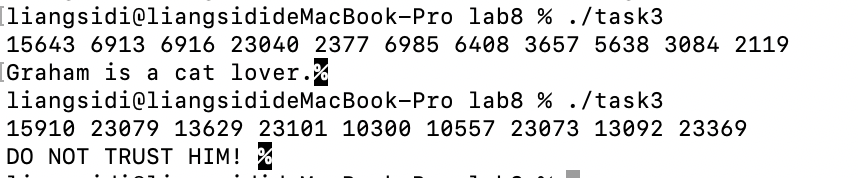


Figure 3: Output of Code in Task 3.