Introductory Programming UESTC1005

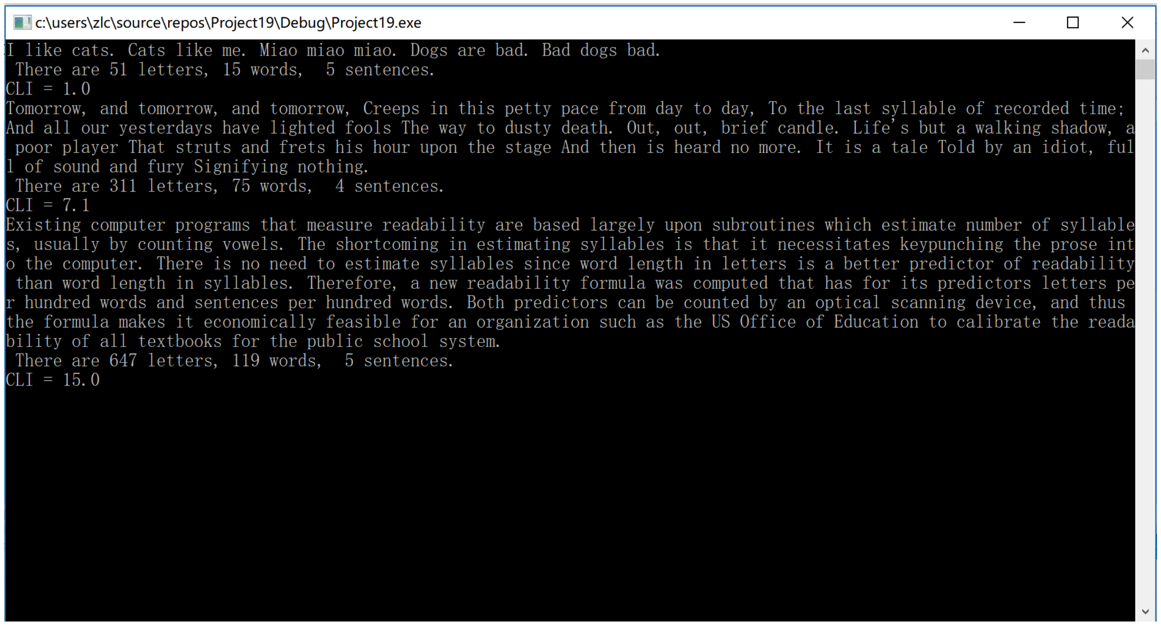
STUDENT NAME:____张立澄

STUDENT NUMBER:____2017200602011

You will need to complete the demonstration of this week lab and submit this report into BB (it will be added into your portfolio of work).

Exercise 12: Strings are Arrays of Characters

Task_A: Follow the instructions in Week 14 Lab manual and past your screen short of your result here.



c:\users\zlc\source\repos\Project19\Debug\Project19.exe

```
I like cats. Cats like me. Miao miao miao. Dogs are bad. Bad dogs bad.
 There are 51 letters, 15 words,  5 sentences.
CLI = 1.0
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, ful
l of sound and fury Signifying nothing.
 There are 311 letters, 75 words,  4 sentences.
CLI = 7.1
Existing computer programs that measure readability are based largely upon subroutines which estimate number of syllable
s, usually by counting vowels. The shortcoming in estimating syllables is that it necessitates keypunching the prose int
o 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 pe
r 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 reada
bility of all textbooks for the public school system.
 There are 647 letters, 119 words,  5 sentences.
CLI = 15.0
```

Task_B: Please attach your source code here.

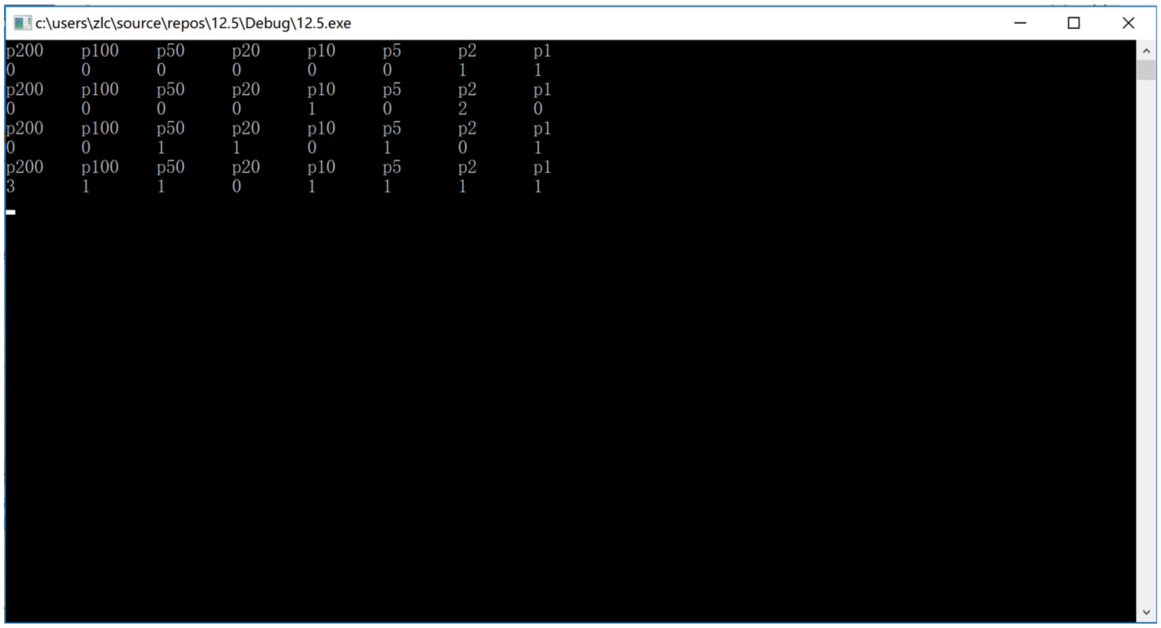
```c
//This is a program that calculates the Coleman-Liau Index of some text.
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<string.h>
int CLI(char text[]) {
        int i;
        int characters, letters = 0, words, spaces = 0, sentences = 0;
        characters = strlen(text);
        for (i = 0; i<characters; i++) {
                if (text[i] == ' ')
                {
                        spaces++;
                }
                if (text[i] == '.')
                {
                        sentences++;
                }
                if ((text[i] >= 'A'&&text[i] <= 'Z') || (text[i] >= 'a'&&text[i] <= 'z'))
                {
                        letters++;
                }
        }
        words = spaces + 1;
        float cli = 5.89 * ((float)letters / words) - 29.5 * ((float)sentences / words) - 15.8;
        //float cli = （ 5.89 * （（float）letters / words) -  29.5*( (float)sentences / words)）  - 15.8;

        if (cli <= 1) {
                cli = 1;
        }
        printf("%s\n", text);
            " There are %d letters, %d words,  %d sentences.\n"
        printf("CLI = %.1f\n", cli);
        return
}

int main() {
        char            "I like cats. Cats like me. Miao miao miao. Dogs are bad. Bad dogs bad."
        char            "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             "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."
        CLI(text1);
        CLI(text2);
        CLI(text3);
        getchar();
}
```

Exercise 13: Combining variables with struct

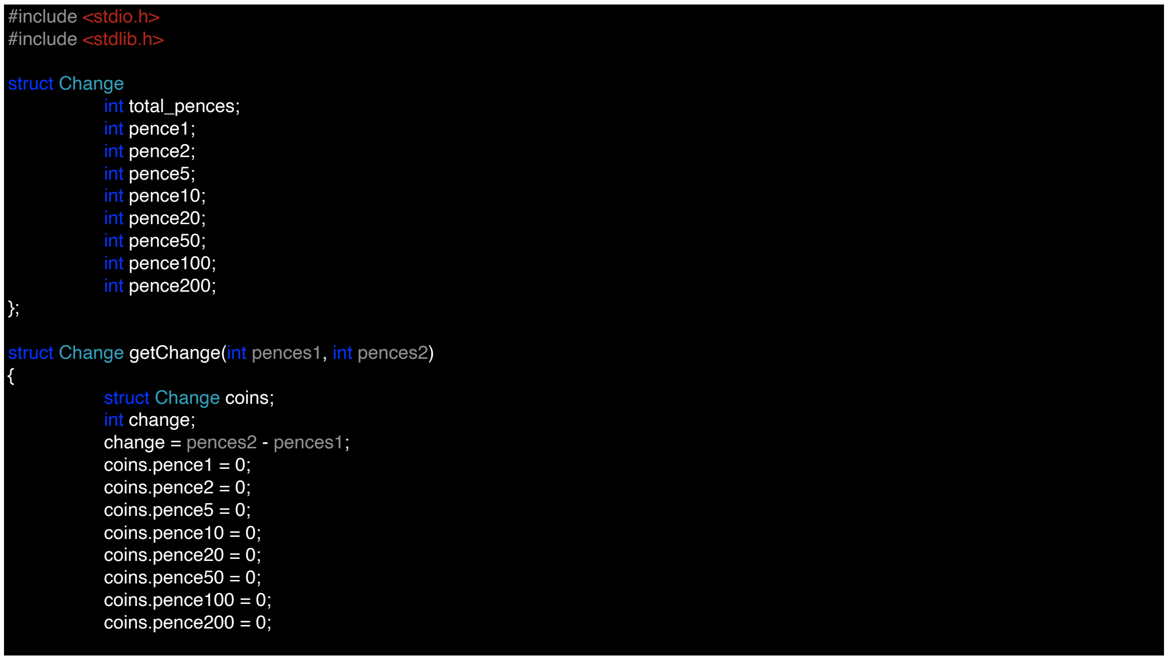
Task_A: Follow the instructions in Week 14 Lab manual and past your screen short of your result here.



c:\users\zlc\source\repos\12.5\Debug\12.5.exe

```
p200    p100    p50     p20     p10     p5      p2      p1
0       0       0       0       0       0       1       1
p200    p100    p50     p20     p10     p5      p2      p1
0       0       0       0       1       0       2       0
p200    p100    p50     p20     p10     p5      p2      p1
0       0       1       1       0       1       0       1
p200    p100    p50     p20     p10     p5      p2      p1
3       1       1       0       1       1       1       1
```

Task_B: Please attach your source code here.



```
#include <stdio.h>
#include <stdlib.h>

struct Change
		int total_pences;
		int pence1;
		int pence2;
		int pence5;
		int pence10;
		int pence20;
		int pence50;
		int pence100;
		int pence200;
};

struct Change getChange(int pences1, int pences2)
{
		struct Change coins;
		int change;
		change = pences2 - pences1;
		coins.pence1 = 0;
		coins.pence2 = 0;
		coins.pence5 = 0;
		coins.pence10 = 0;
		coins.pence20 = 0;
		coins.pence50 = 0;
		coins.pence100 = 0;
		coins.pence200 = 0;
```

```
        if (change >= 200)
        {
                coins.pence200 = change / 200;
                change = change - coins.pence200 * 200;
        }
        if (change >= 100)
        {
                coins.pence100 = change / 100;
                change = change - coins.pence100 * 100;
        }
        if (change >= 50)
        {
                coins.pence50 = change / 50;
                change = change - coins.pence50 * 50;
        }
        if (change >= 20)
        {
                coins.pence20 = change / 20;
                change = change - coins.pence20 * 20;
        }
        if (change >= 10)
        {
                coins.pence10 = change / 10;
                change = change - coins.pence10 * 10;
        }
        if (change >= 5)
        {
                coins.pence5 = change / 5;
                change = change - coins.pence5 * 5;
        }
        if (change >= 2)
        {
                coins.pence2 = change / 2;
                change = change - coins.pence2 * 2;
        }
        if (change == 1)
        {
                coins.pence1 = change / 1;
                change = change - coins.pence1 * 1;
        }
        return coins;
}

void printChange(struct Change coins)
{
            "p200\tp100\tp50\tp20\tp10\tp5\tp2\tp1\n"
        printf("%i\t%i\t%i\t%i\t%i\t%i\t%i\t%i\n", coins.pence200, coins.pence100, coins.pence50, coins.pence20, coins.pence10,
coins.pence5, coins.pence2, coins.pence1);

}


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();
}
}
```