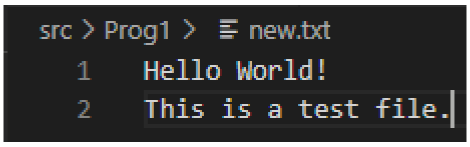
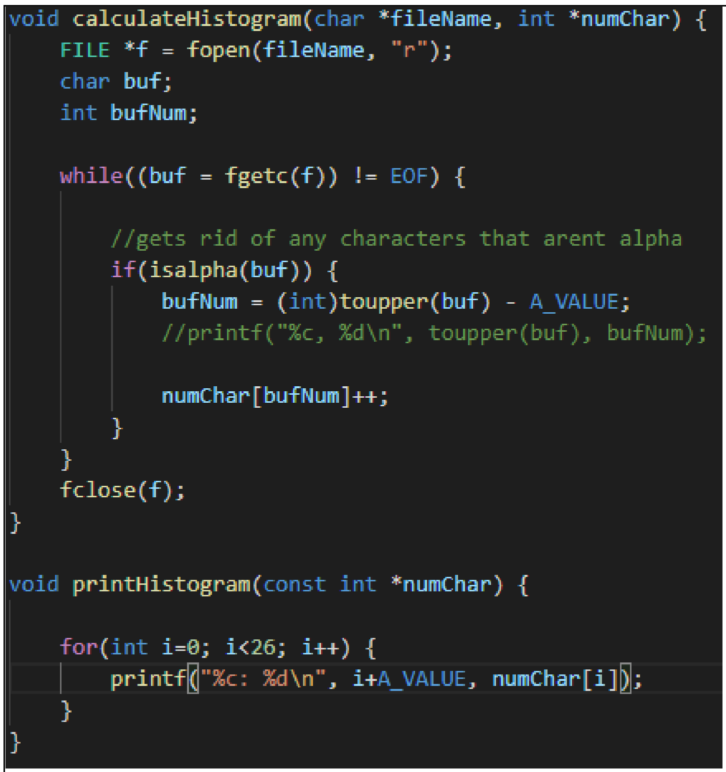
C4 – Pointers, Files and Strings

3.1 Reading Files

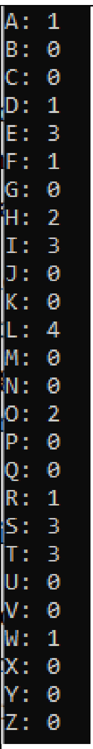
Test text file



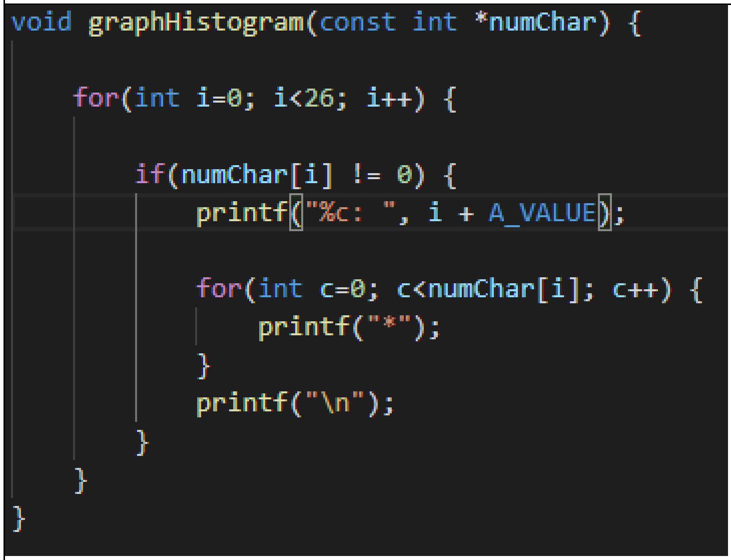
```
src > Prog1 > new.txt
Hello World!
This is a test file.
```



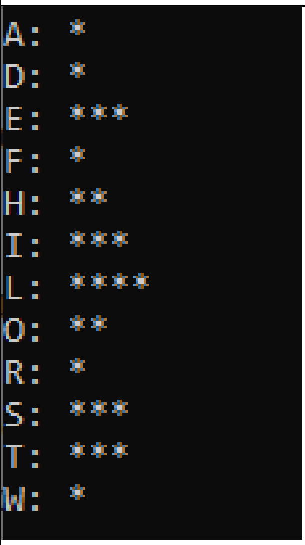
```c
void calculateHistogram(char *fileName, int *numChar) {
    FILE *f = fopen(fileName, "r");
    char buf;
    int bufNum;

    while((buf = fgetc(f)) != EOF) {

        //gets rid of any characters that arent alpha
        if(isalpha(buf)) {
            bufNum = (int)toupper(buf) - A_VALUE;
            //printf("%c, %d\n", toupper(buf), bufNum);

            numChar[bufNum]++;
        }
    }
    fclose(f);
}

void printHistogram(const int *numChar) {

    for(int i=0; i<26; i++) {
        printf("%c: %d\n", i+A_VALUE, numChar[i]);
    }
}
```



```
A: 1
B: 0
C: 0
D: 1
E: 3
F: 1
G: 0
H: 2
I: 3
J: 0
K: 0
L: 4
M: 0
N: 0
O: 2
P: 0
Q: 0
R: 1
S: 3
T: 3
U: 0
V: 0
W: 1
X: 0
Y: 0
Z: 0
```

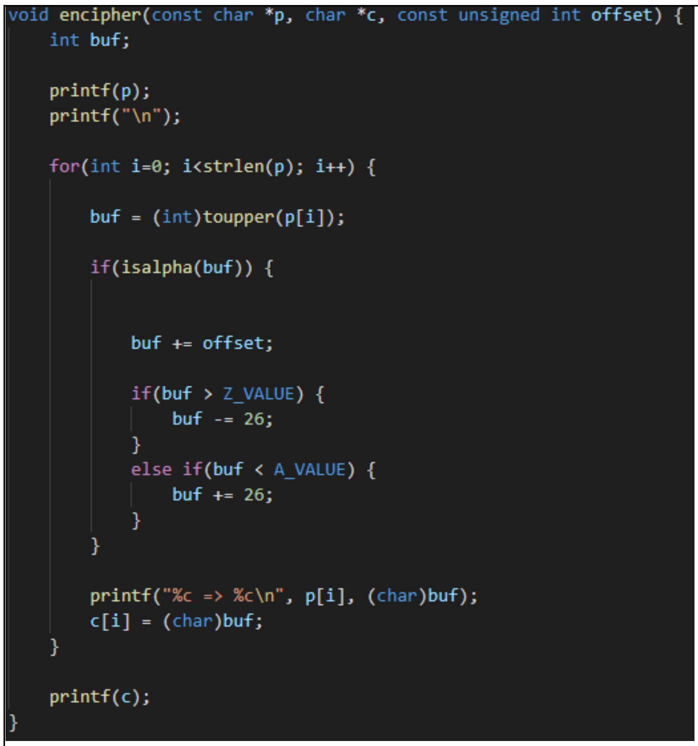


```c
void graphHistogram(const int *numChar) {

    for(int i=0; i<26; i++) {

        if(numChar[i] != 0) {
            printf("%c: ", i + A_VALUE);

            for(int c=0; c<numChar[i]; c++) {
                printf("*");
            }
            printf("\n");
        }
    }
}
```

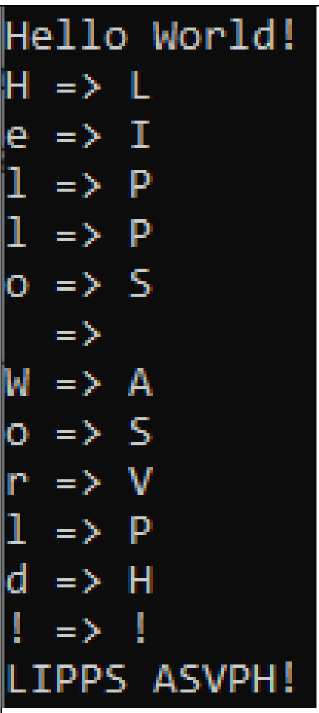


```
A: *
D: *
E: ***
F: *
H: **
I: ***
L: ****
O: **
R: *
S: ***
T: ***
W: *
```

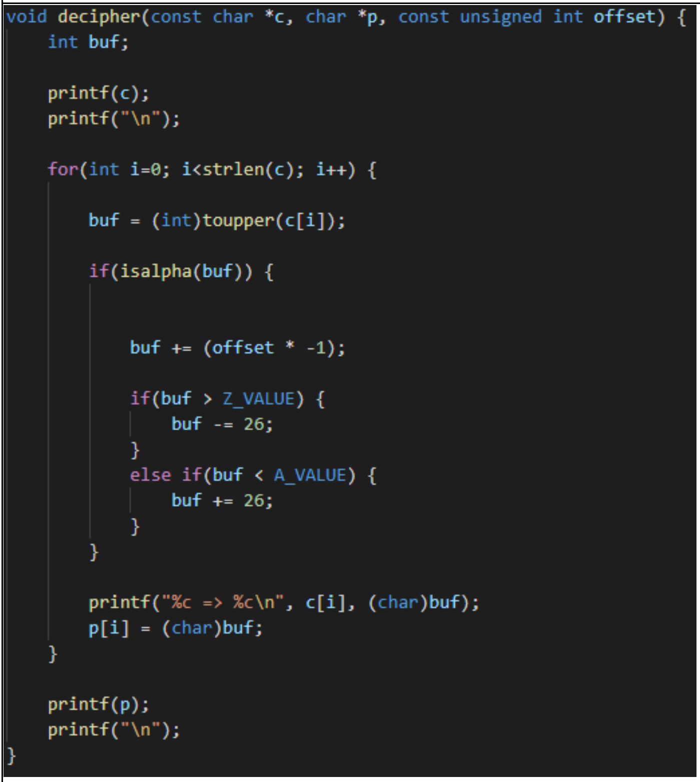
3.2 Manipulation Strings



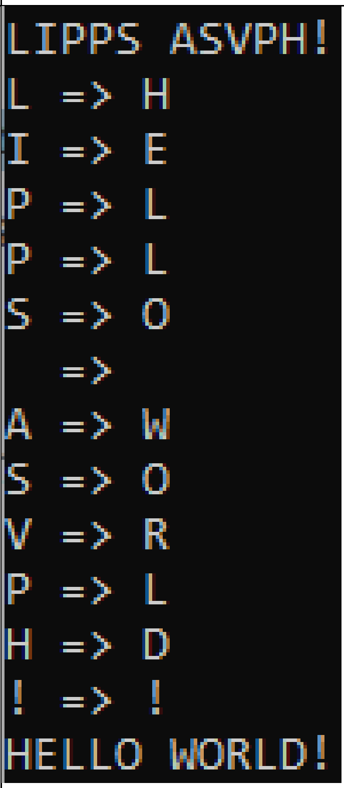
```c
void encipher(const char *p, char *c, const unsigned int offset) {
    int buf;

    printf(p);
    printf("\n");

    for(int i=0; i<strlen(p); i++) {

        buf = (int)toupper(p[i]);

        if(isalpha(buf)) {


            buf += offset;

            if(buf > Z_VALUE) {
                buf -= 26;
            }
            else if(buf < A_VALUE) {
                buf += 26;
            }
        }

        printf("%c => %c\n", p[i], (char)buf);
        c[i] = (char)buf;
    }

    printf(c);
}
```



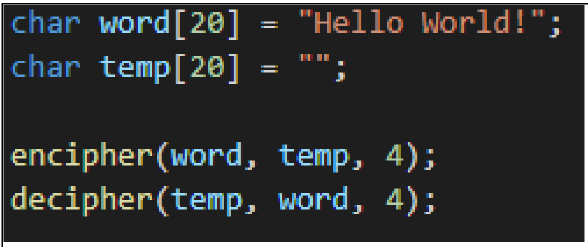
```
Hello World!
H => L
e => I
l => P
l => P
o => S
  =>
W => A
o => S
r => V
l => P
d => H
! => !
LIPPS ASVPH!
```



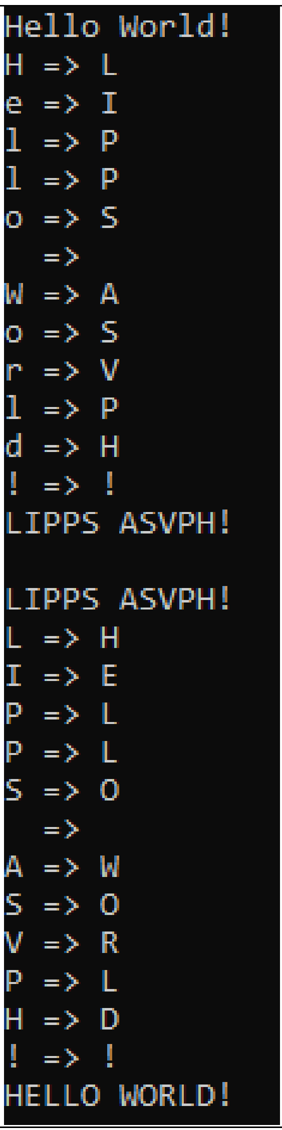
```c
void decipher(const char *c, char *p, const unsigned int offset) {
    int buf;

    printf(c);
    printf("\n");

    for(int i=0; i<strlen(c); i++) {

        buf = (int)toupper(c[i]);

        if(isalpha(buf)) {


            buf += (offset * -1);

            if(buf > Z_VALUE) {
                buf -= 26;
            }
            else if(buf < A_VALUE) {
                buf += 26;
            }
        }

        printf("%c => %c\n", c[i], (char)buf);
        p[i] = (char)buf;
    }

    printf(p);
    printf("\n");
}
```



```
LIPPS ASVPH!
L => H
I => E
P => L
P => L
S => O
  =>
A => W
S => O
V => R
P => L
H => D
! => !
HELLO WORLD!
```

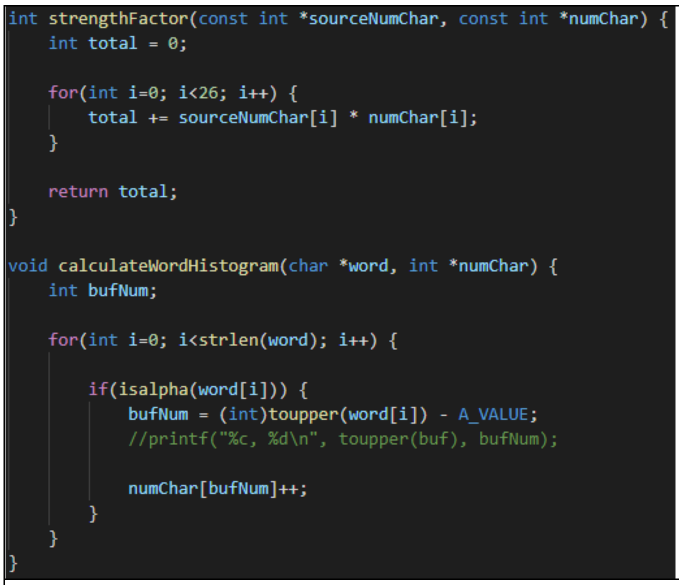


```c
char word[20] = "Hello World!";
char temp[20] = "";

encipher(word, temp, 4);
decipher(temp, word, 4);
```



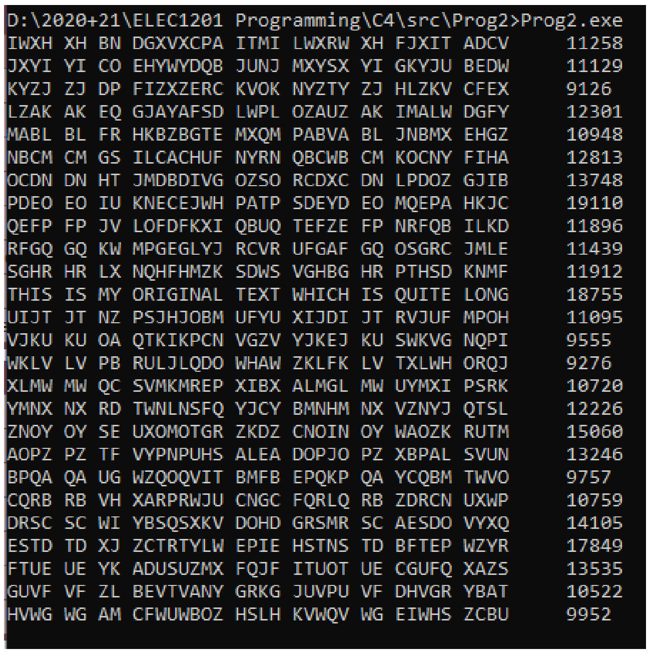
```
Hello World!
H => L
e => I
l => P
l => P
o => S
  =>
W => A
o => S
r => V
l => P
d => H
! => !
LIPPS ASVPH!

LIPPS ASVPH!
L => H
I => E
P => L
P => L
S => O
  =>
A => W
S => O
V => R
P => L
H => D
! => !
HELLO WORLD!
```

3.3 Code Breaking



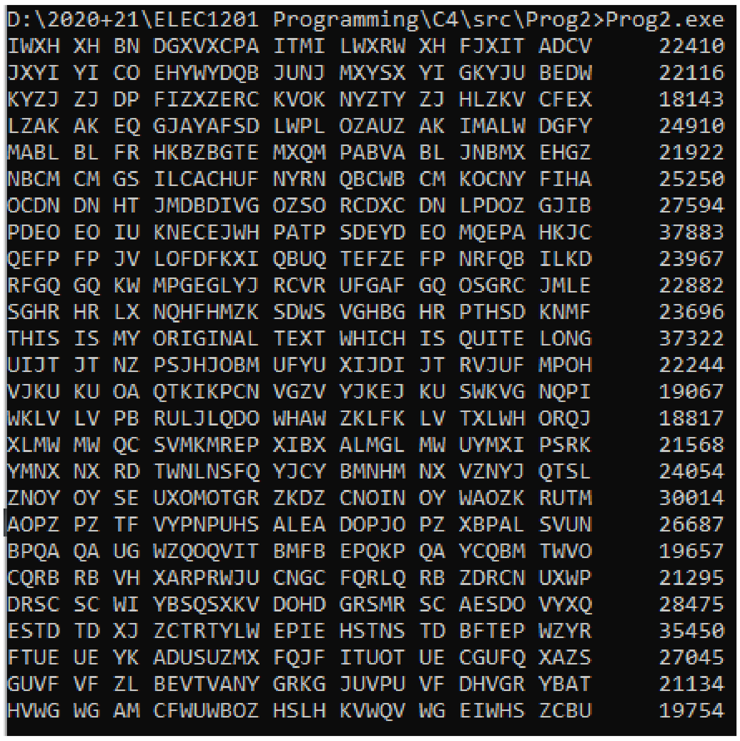
```c
int strengthFactor(const int *sourceNumChar, const int *numChar) {
    int total = 0;

    for(int i=0; i<26; i++) {
        total += sourceNumChar[i] * numChar[i];
    }

    return total;
}

void calculateWordHistogram(char *word, int *numChar) {
    int bufNum;

    for(int i=0; i<strlen(word); i++) {

        if(isalpha(word[i])) {
            bufNum = (int)toupper(word[i]) - A_VALUE;
            //printf("%c, %d\n", toupper(buf), bufNum);

            numChar[bufNum]++;
        }
    }
}
```

I have now used a text file which has a lot of text in (source http://randomtextgenerator.com/) which then generated its own histogram for each letter. Then I take the text to be deciphered and sum the product of each of its histogram against the source histogram (e.g. a * a, …). Then it repeats, changing the increment each time. In theory the letters which are most common in both will yield a higher total "strength factor" and are therefore a possible greater match. In practice, the original correct text is usually one of the highest ones but not always the highest as seen below.



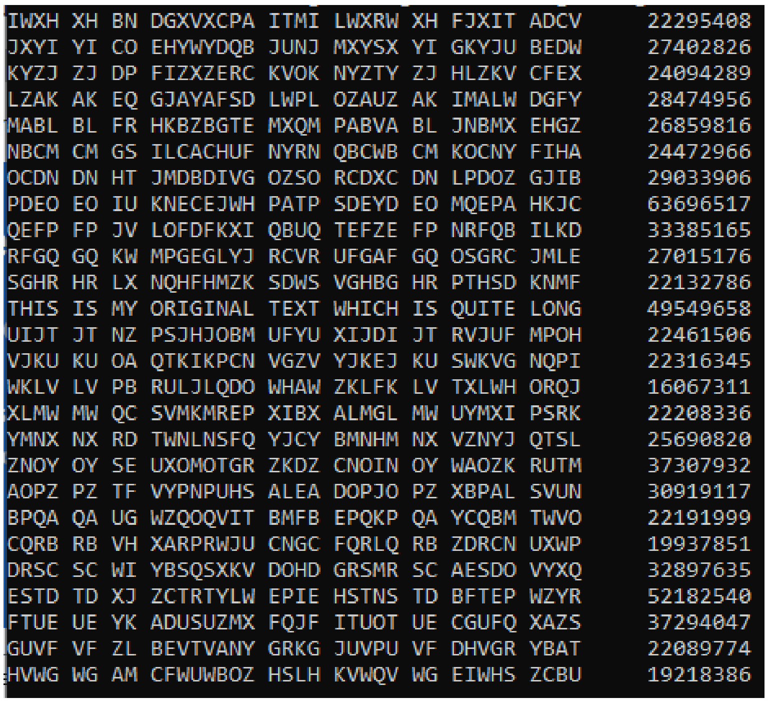
```
D:\2020+21\ELEC1201 Programming\C4\src\Prog2>Prog2.exe
IWXH XH BN DGXVXCPA ITMI LWXRW XH FJXIT ADCV      11258
JXYI YI CO EHYWYDQB JUNJ MXYSX YI GKYJU BEDW      11129
KYZJ ZJ DP FIZXZERC KVOK NYZTY ZJ HLZKV CFEX      9126
LZAK AK EQ GJAYAFSD LWPL OZAUZ AK IMALW DGFY      12301
MABL BL FR HKBZBGTE MXQM PABVA BL JNBMX EHGZ      10948
NBCM CM GS ILCACHUF NYRN QBCWB CM KOCNY FIHA      12813
OCDN DN HT JMDBDIVG OZSO RCDXC DN LPDOZ GJIB      13748
PDEO EO IU KNECEJWH PATP SDEYD EO MQEPA HKJC      19110
QEFP FP JV LOFDFKXI QBUQ TEFZE FP NRFQB ILKD      11896
RFGQ GQ KW MPGEGLYJ RCVR UFGAF GQ OSGRC JMLE      11439
SGHR HR LX NQHFHMZK SDWS VGHBG HR PTHSD KNMF      11912
THIS IS MY ORIGINAL TEXT WHICH IS QUITE LONG      18755
UIJT JT NZ PSJHJOBM UFYU XIJDI JT RVJUF MPOH      11095
VJKU KU OA QTKIKPCN VGZV YJKEJ KU SWKVG NQPI      9555
WKLV LV PB RULJLQDO WHAW ZKLFK LV TXLWH ORQJ      9276
XLMW MW QC SVMKMREP XIBX ALMGL MW UYMXI PSRK      10720
YMNX NX RD TWNLNSFQ YJCY BMNHM NX VZNYJ QTSL      12226
ZNOY OY SE UXOMOTGR ZKDZ CNOIN OY WAOZK RUTM      15060
AOPZ PZ TF VYPNPUHS ALEA DOPJO PZ XBPAL SVUN      13246
BPQA QA UG WZQOQVIT BMFB EPQKP QA YCQBM TWVO      9757
CQRB RB VH XARPRWJU CNGC FQRLQ RB ZDRCN UXWP      10759
DRSC SC WI YBSQSXKV DOHD GRSMR SC AESDO VYXQ      14105
ESTD TD XJ ZCTRTYLW EPIE HSTNS TD BFTEP WZYR      17849
FTUE UE YK ADUSUZMX FQJF ITUOT UE CGUFQ XAZS      13535
GUVF VF ZL BEVTVANY GRKG JUVPU VF DHVGR YBAT      10522
HVWG WG AM CFWUWBOZ HSLH KVWQV WG EIWHS ZCBU      9952
```

After I had increased the size of the source text (doubled in size using text from the same website as before) this created a greater differences of bad matches and good matches but my original sentence still didn't come out as the highest but by a very small margin of 37322 compared to the highest which was 37883.



```
D:\2020+21\ELEC1201 Programming\C4\src\Prog2>Prog2.exe
IWXH XH BN DGXVXCPA ITMI LWXRW XH FJXIT ADCV      22410
JXYI YI CO EHYWYDQB JUNJ MXYSX YI GKYJU BEDW      22116
KYZJ ZJ DP FIZXZERC KVOK NYZTY ZJ HLZKV CFEX      18143
LZAK AK EQ GJAYAFSD LWPL OZAUZ AK IMALW DGFY      24910
MABL BL FR HKBZBGTE MXQM PABVA BL JNBMX EHGZ      21922
NBCM CM GS ILCACHUF NYRN QBCWB CM KOCNY FIHA      25250
OCDN DN HT JMDBDIVG OZSO RCDXC DN LPDOZ GJIB      27594
PDEO EO IU KNECEJWH PATP SDEYD EO MQEPA HKJC      37883
QEFP FP JV LOFDFKXI QBUQ TEFZE FP NRFQB ILKD      23967
RFGQ GQ KW MPGEGLYJ RCVR UFGAF GQ OSGRC JMLE      22882
SGHR HR LX NQHFHMZK SDWS VGHBG HR PTHSD KNMF      23696
THIS IS MY ORIGINAL TEXT WHICH IS QUITE LONG      37322
UIJT JT NZ PSJHJOBM UFYU XIJDI JT RVJUF MPOH      22244
VJKU KU OA QTKIKPCN VGZV YJKEJ KU SWKVG NQPI      19067
WKLV LV PB RULJLQDO WHAW ZKLFK LV TXLWH ORQJ      18817
XLMW MW QC SVMKMREP XIBX ALMGL MW UYMXI PSRK      21568
YMNX NX RD TWNLNSFQ YJCY BMNHM NX VZNYJ QTSL      24054
ZNOY OY SE UXOMOTGR ZKDZ CNOIN OY WAOZK RUTM      30014
AOPZ PZ TF VYPNPUHS ALEA DOPJO PZ XBPAL SVUN      26687
BPQA QA UG WZQOQVIT BMFB EPQKP QA YCQBM TWVO      19657
CQRB RB VH XARPRWJU CNGC FQRLQ RB ZDRCN UXWP      21295
DRSC SC WI YBSQSXKV DOHD GRSMR SC AESDO VYXQ      28475
ESTD TD XJ ZCTRTYLW EPIE HSTNS TD BFTEP WZYR      35450
FTUE UE YK ADUSUZMX FQJF ITUOT UE CGUFQ XAZS      27045
GUVF VF ZL BEVTVANY GRKG JUVPU VF DHVGR YBAT      21134
HVWG WG AM CFWUWBOZ HSLH KVWQV WG EIWHS ZCBU      19754
```

Next I created a new function which can square the histogram of the source text so that there is a greater difference between the popular and unpopular characters. Unfortunately this made a larger difference between the false positive result and the official result as show below.



```
IWXH XH BN DGXVXCPA ITMI LWXRW XH FJXIT ADCV      22295408
JXYI YI CO EHYWYDQB JUNJ MXYSX YI GKYJU BEDW      27402826
KYZJ ZJ DP FIZXZERC KVOK NYZTY ZJ HLZKV CFEX      24094289
LZAK AK EQ GJAYAFSD LWPL OZAUZ AK IMALW DGFY      28474956
MABL BL FR HKBZBGTE MXQM PABVA BL JNBMX EHGZ      26859816
NBCM CM GS ILCACHUF NYRN QBCWB CM KOCNY FIHA      24472966
OCDN DN HT JMDBDIVG OZSO RCDXC DN LPDOZ GJIB      29033906
PDEO EO IU KNECEJWH PATP SDEYD EO MQEPA HKJC      63696517
QEFP FP JV LOFDFKXI QBUQ TEFZE FP NRFQB ILKD      33385165
RFGQ GQ KW MPGEGLYJ RCVR UFGAF GQ OSGRC JMLE      27015176
SGHR HR LX NQHFHMZK SDWS VGHBG HR PTHSD KNMF      22132786
THIS IS MY ORIGINAL TEXT WHICH IS QUITE LONG      49549658
UIJT JT NZ PSJHJOBM UFYU XIJDI JT RVJUF MPOH      22461506
VJKU KU OA QTKIKPCN VGZV YJKEJ KU SWKVG NQPI      22316345
WKLV LV PB RULJLQDO WHAW ZKLFK LV TXLWH ORQJ      16067311
XLMW MW QC SVMKMREP XIBX ALMGL MW UYMXI PSRK      22208336
YMNX NX RD TWNLNSFQ YJCY BMNHM NX VZNYJ QTSL      25690820
ZNOY OY SE UXOMOTGR ZKDZ CNOIN OY WAOZK RUTM      37307932
AOPZ PZ TF VYPNPUHS ALEA DOPJO PZ XBPAL SVUN      30919117
BPQA QA UG WZQOQVIT BMFB EPQKP QA YCQBM TWVO      22191999
CQRB RB VH XARPRWJU CNGC FQRLQ RB ZDRCN UXWP      19937851
DRSC SC WI YBSQSXKV DOHD GRSMR SC AESDO VYXQ      32897635
ESTD TD XJ ZCTRTYLW EPIE HSTNS TD BFTEP WZYR      52182540
FTUE UE YK ADUSUZMX FQJF ITUOT UE CGUFQ XAZS      37294047
GUVF VF ZL BEVTVANY GRKG JUVPU VF DHVGR YBAT      22089774
HVWG WG AM CFWUWBOZ HSLH KVWQV WG EIWHS ZCBU      19218386
```

Therefore in conclusion to make the results more accurate the text file should be as long as possible to create the greatest accuracy of what English text best consists of. The program should also output the best 3 or 4 matches since the “best match” is usually not the actual answer.