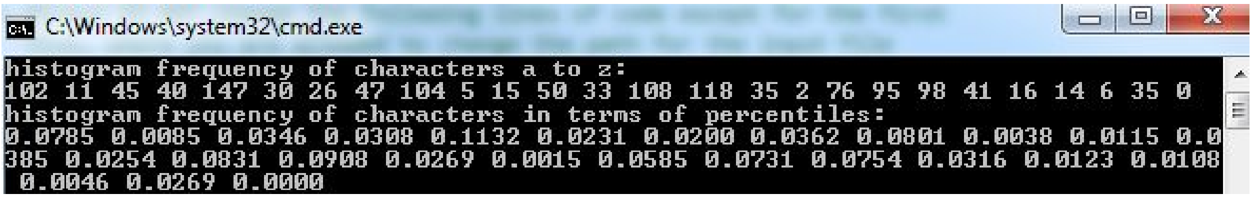
Final project : Histogram

a) Running the CPU version of the code :



```
histogram frequency of characters a to z:
102 11 45 40 147 30 26 47 104 5 15 50 33 108 118 35 2 76 95 98 41 16 14 6 35 0
histogram frequency of characters in terms of percentiles:
0.0785 0.0085 0.0346 0.0308 0.1132 0.0231 0.0200 0.0362 0.0801 0.0038 0.0115 0.0
385 0.0254 0.0831 0.0908 0.0269 0.0015 0.0585 0.0731 0.0754 0.0316 0.0123 0.0108
 0.0046 0.0269 0.0000
```

b) Code after modification :

```
#include "J:\ami\ami\common\book.h"
#include <cuda_runtime_api.h>
#include "device_launch_parameters.h"
#include <stdio.h>
#include <cuda.h>
#include <stdlib.h>
#include <time.h>
#include "..\..\ami\ami\common\cpu_bitmap.h"
#include<math.h>
#include "J:\ami\ami\common\cpu_anim.h"


/*
*
* Author:      Brady Chen  4/12/2017
* Modified By:
*              <Amikar Divij> <5/5/2017>
*
* This is a C code for the computation of a histogram of data from an input text file. The
* text file contains multiple lines of characters. The code generate the frequency histogram
* of characters from the input file.
*/


/*
*
* Definition of important values:
*   MAX_TEXT_LINES  -- maximum number of lines for the input file. We initially set to 1000
*                      If your file contains more than 1000, just change the value
*   MAX_LINE_LENGTH -- the length of each line. You change the value
*   NUMBER_OF_CHARS -- this the total number of characters used in the input file. We set
```

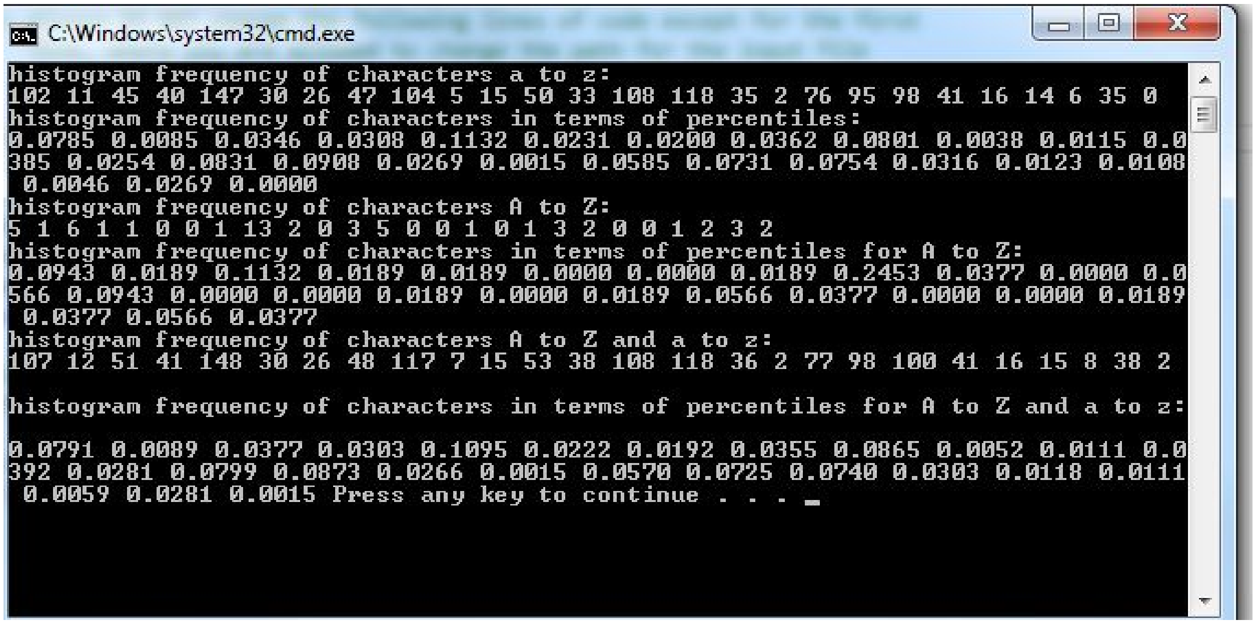
```
    *                   the number to 128 which could include all the standard characters in
    *                   the ASCII code table. If your input file contains more than 128 standard
    *                   characters, please increase the number. For example, you should use 256
    *                   for extended ASCII code.
    *   TOTAL_SIZE      -- this is the total number of characters in the input file.
    *
    * NOTE: You can change the values of the varibales if necessary but please don't change
the variable
    *       names themselves
    */
    #define MAX_TEXT_LINES 4400
    #define MAX_LINE_LENGTH 75
    #define NUMBER_OF_CHARS 128
    #define TOTAL_SIZE (MAX_TEXT_LINES + 10)*(MAX_LINE_LENGTH + 10)

    /*
    * Declaraion of the arrays of characters
    *   aTextData    -- this is a two dimensional array of char. It contains all the characters
    *                 inputted from the input file. The first index record the lines and the
    *                 second index record the columns in each line. We add 10 on
MAX_TEXT_LINES
    *                 and MAX_LINE_LENGTH as safety spaces to crash on off-by-one errors.
    *
    *   NOTE: No need to change this part
    */
    char aTextData[MAX_TEXT_LINES + 10][MAX_LINE_LENGTH + 10];
    char aFlattenedData[(MAX_TEXT_LINES + 10)*(MAX_LINE_LENGTH + 10)];


    /*
    *
    * Declaration of function
    *   histogram() -- This function takes two arguments
    *                aText -- a two dimensional array of char.
    *                histo -- a one dimensional array which returns the frequency histogram
    *                       of the characrers in the array aText
    */
    __global__ void gpu_histogram(unsigned char *aText, unsigned int *histo);

    int main(void)
    {
        /*
        *  Declaration of variables
        *    histo       -- contains the frequency histogram of the characters
        *    lineBuffer -- line buffer for reading
        *    iTextSize  -- how large is the array
        *    iTextUsed  -- how many lines contained
        */
        unsigned int histo[NUMBER_OF_CHARS];
        char lineBuffer[MAX_LINE_LENGTH + 10];
        int  iTextSize = MAX_TEXT_LINES;
        int  iTextUsed = 0;

        // clean the memory with zeros
        memset(aTextData, 0, sizeof(aTextData));
        memset(lineBuffer, 0, sizeof(lineBuffer));

        /*
        *  Data read and conversion: The following lines of code do the following
        *    1. Find the input file (in this case "input.txt)
        *    2. Read the data from the input file and store it in a two dimensional
        *       array of char
        *
        *
        */

        char *inputFile = "J:/FSD/FSD/input_test.txt"; // you may need to change the path
depending on
        // where you put the input file
        // read input file into the array. use binary mode "rb"
        FILE *fin = fopen(inputFile, "rb");
        if (!fin) {
            printf("cannot read %s\n", inputFile);
            return 1;
        }
        // while not end of file, read another line
        while (fgets(lineBuffer, sizeof(lineBuffer)-10, fin) != 0)
        {
            // check if array has space for another line
            if (iTextUsed >= iTextSize)
            {
                printf("overflow: too many text lines\n");
                return 1;
            }
            // strip CR/LF from line endings so we get pure text
            char *psz = strchr(lineBuffer, '\r'); if (psz) *psz = '\0';
            psz = strchr(lineBuffer, '\n'); if (psz) *psz = '\0';
            // store the line
            strncpy(aTextData[iTextUsed], lineBuffer, MAX_LINE_LENGTH);
            aTextData[iTextUsed][MAX_LINE_LENGTH - 1] = '\0'; // safety
            iTextUsed++;
        }
        fclose(fin);
        /* end of the Data read and conversion */



        /*
        * Function call to compute the frequency histogram of the data
        *       1. declare’s the blocks and threads
        *       2. call’s the implemented kernel function gpu_histogram() .
        *       Please note that the kernel function DOES NOT take array as argument,
        *       not to mention two dimensional array in this case.
        */
        unsigned int *histo1;

        unsigned char *aText;
        cudaMalloc((void**)&aText, TOTAL_SIZE);
        cudaMemcpy(aText, aTextData, TOTAL_SIZE,
            cudaMemcpyHostToDevice);

        cudaMalloc((void**)&histo1,
            NUMBER_OF_CHARS * sizeof(int));
        cudaMemset(histo1, 0,
            NUMBER_OF_CHARS * sizeof(int));

        // kernel launch - 2x the number of mps gave best timing
        cudaDeviceProp  prop;
        cudaGetDeviceProperties(&prop, 0);
        unsigned int blocks = prop.multiProcessorCount;
        gpu_histogram << <blocks * 2, NUMBER_OF_CHARS >> >(aText, histo1);



        cudaMemcpy(histo, histo1,
            NUMBER_OF_CHARS * sizeof(int),
            cudaMemcpyDeviceToHost);
```

```
    /*
    *
    * Output of histogram results
    * the following is the output of the histogram frequency of characters in
    * the order of ASCII code. the array histo[] contains the frequency histogram
    * of 128 characters defined in ASCII.
    *   1. The decimal values for the letters are
    *      A to Z -- 65 to 90
    *      a to z -- 97 to 122
    *   2. the following lines of code print the histogram results for a to z, A to   * Z. and both
lower and upper case a to z with their percentiles.
    *
    *
    */
    unsigned int histocount = 0; // total character count
    printf("histogram frequency of characters a to z: \n");
    for (int i = 97; i < 97 + 26; i++) {
        histocount += histo[i];
        printf("%d ", histo[i]);
    }
    printf("\n");
    printf("histogram frequency of characters in terms of percentiles: \n");
    for (int i = 97; i < 97 + 26; i++) {
        printf("%.4f ", (float)histo[i] / (float)histocount);
    }

    printf("\n");

    unsigned int histocount1 = 0; // total character count
    printf("histogram frequency of characters A to Z: \n");
    for (int i = 65; i < 65 + 26; i++) {
        histocount1 += histo[i];
        printf("%d ", histo[i]);
    }
    printf("\n");
    printf("histogram frequency of characters in terms of percentiles for A to Z: \n");
    for (int i = 65; i < 65 + 26; i++) {
        printf("%.4f ", (float)histo[i] / (float)histocount1);
    }
    printf("\n");

    unsigned int histocount2 = 0;
```

```
        printf("histogram frequency of characters A to Z and a to z: \n");

        for (int i = 0; i < 0 + 26; i++) {
                histocount2 += histo[65 + i];

                printf("%d ", histo[65+i]+histo[97+i]);

        }
        printf("\n");
        unsigned int histocount3 = 0;
        histocount3 = histocount + histocount1;

        printf("histogram frequency of characters in terms of percentiles for A to Z and a to z:
\n");
        for (int i = 0; i < 0 + 26; i++) {
                printf("%.4f ", ((float)histo[65 + i] + (float)histo[97 + i] ) / (float)histocount3);

        }

        /* code above here for computing the frequency histogram for all letters
        *       regardless low or up cases.
        *
        */
        cudaFree(histo);
        cudaFree(aText);
        cudaFree(aTextData);

        return 0;
    }

    /*
    *
    * Implementation of function
    *   histogram() -- This function takes two pointer arguments
    *               aText -- a two dimensional array of char.
    *               histo -- a one dimensional array which returns the frequency histogram
    *                       of the characrers in the array aText
    *it used atomicadd to perform the increment function without causing any error due to
*different threads accessing the same variable
    */

    __global__ void gpu_histogram(unsigned char *aText, unsigned int *histo)
    {
```

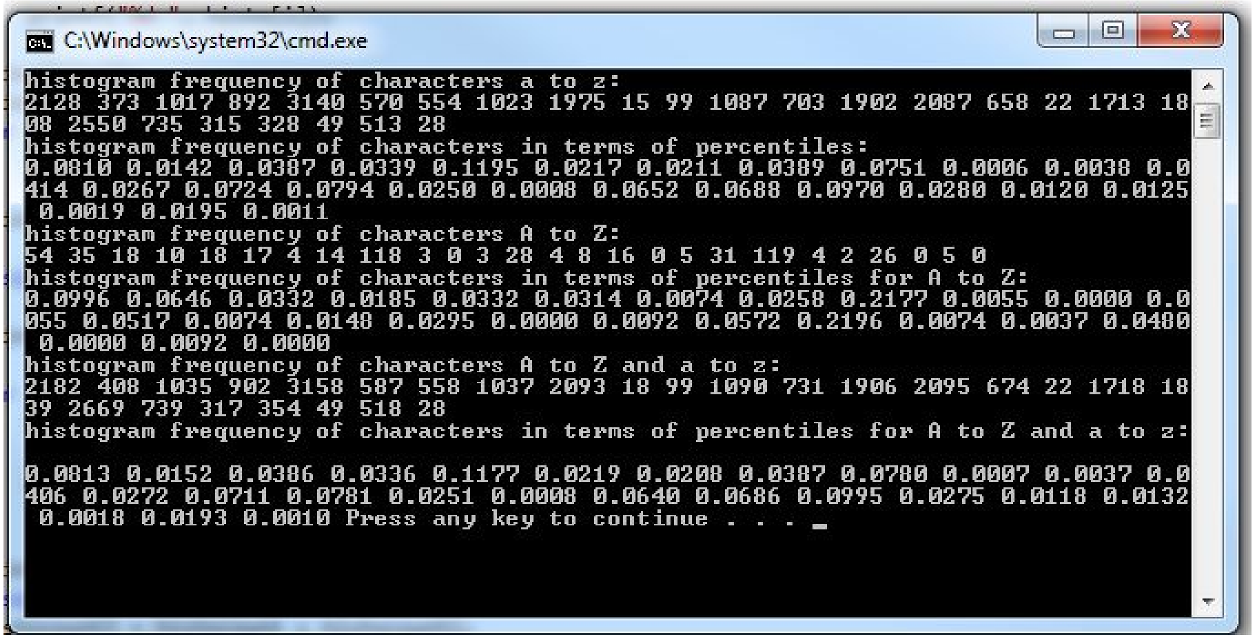
```
    int i = threadIdx.x + blockIdx.x * blockDim.x;
    int stride = blockDim.x * gridDim.x;


    while (i < TOTAL_SIZE){
            atomicAdd(&(histo[aText[i]]), 1);
            i += stride;
    }
}
```

c) Result of GPU histogram on input_test



```
histogram frequency of characters a to z:
102 11 45 40 147 30 26 47 104 5 15 50 33 108 118 35 2 76 95 98 41 16 14 6 35 0
histogram frequency of characters in terms of percentiles:
0.0785 0.0085 0.0346 0.0308 0.1132 0.0231 0.0200 0.0362 0.0801 0.0038 0.0115 0.0
385 0.0254 0.0831 0.0908 0.0269 0.0015 0.0585 0.0731 0.0754 0.0316 0.0123 0.0108
 0.0046 0.0269 0.0000
histogram frequency of characters A to Z:
5 1 6 1 1 0 0 1 13 2 0 3 5 0 0 1 0 1 3 2 0 0 1 2 3 2
histogram frequency of characters in terms of percentiles for A to Z:
0.0943 0.0189 0.1132 0.0189 0.0189 0.0000 0.0000 0.0189 0.2453 0.0377 0.0000 0.0
566 0.0943 0.0000 0.0000 0.0189 0.0000 0.0189 0.0566 0.0377 0.0000 0.0000 0.0189
 0.0377 0.0566 0.0377
histogram frequency of characters A to Z and a to z:
107 12 51 41 148 30 26 48 117 7 15 53 38 108 118 36 2 77 98 100 41 16 15 8 38 2

histogram frequency of characters in terms of percentiles for A to Z and a to z:

0.0791 0.0089 0.0377 0.0303 0.1095 0.0222 0.0192 0.0355 0.0865 0.0052 0.0111 0.0
392 0.0281 0.0799 0.0873 0.0266 0.0015 0.0570 0.0725 0.0740 0.0303 0.0118 0.0111
 0.0059 0.0281 0.0015 Press any key to continue . . .
```

d) Result of GPU histogram on input_data



```
histogram frequency of characters a to z:
2128 373 1017 892 3140 570 554 1023 1975 15 99 1087 703 1902 2087 658 22 1713 18
08 2550 735 315 328 49 513 28
histogram frequency of characters in terms of percentiles:
0.0810 0.0142 0.0387 0.0339 0.1195 0.0217 0.0211 0.0389 0.0751 0.0006 0.0038 0.0
414 0.0267 0.0724 0.0794 0.0250 0.0008 0.0652 0.0688 0.0970 0.0280 0.0120 0.0125
 0.0019 0.0195 0.0011
histogram frequency of characters A to Z:
54 35 18 10 18 17 4 14 118 3 0 3 28 4 8 16 0 5 31 119 4 2 26 0 5 0
histogram frequency of characters in terms of percentiles for A to Z:
0.0996 0.0646 0.0332 0.0185 0.0332 0.0314 0.0074 0.0258 0.2177 0.0055 0.0000 0.0
055 0.0517 0.0074 0.0148 0.0295 0.0000 0.0092 0.0572 0.2196 0.0074 0.0037 0.0480
 0.0000 0.0092 0.0000
histogram frequency of characters A to Z and a to z:
2182 408 1035 902 3158 587 558 1037 2093 18 99 1090 731 1906 2095 674 22 1718 18
39 2669 739 317 354 49 518 28
histogram frequency of characters in terms of percentiles for A to Z and a to z:

0.0813 0.0152 0.0386 0.0336 0.1177 0.0219 0.0208 0.0387 0.0780 0.0007 0.0037 0.0
406 0.0272 0.0711 0.0781 0.0251 0.0008 0.0640 0.0686 0.0995 0.0275 0.0118 0.0132
 0.0018 0.0193 0.0010 Press any key to continue . . .
```