

```

////////////////////////////////////
/// 640x480 version!
/// test VGA with hardware video input copy to VGA
// compile with
// gcc fp_test_1.c -o fp1 -lm
////////////////////////////////////
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/mman.h>
#include <sys/time.h>
#include <math.h>

#include "address_map_arm_brl4.h"

#define PIO_RST 0x10
#define PIO_CI1_NEW 0x20
#define PIO_CI2_NEW 0x30
#define PIO_CI3_NEW 0x40

volatile unsigned int * pio_rst_ptr = NULL;
volatile unsigned int * pio_ci1_new_ptr = NULL;
volatile unsigned int * pio_ci2_new_ptr = NULL;
volatile unsigned int * pio_ci3_new_ptr = NULL;

/* function prototypes */
void VGA_text (int, int, char *);
void VGA_text_clear();
void VGA_box (int, int, int, int, short);
void VGA_line(int, int, int, int, short) ;
void VGA_disc (int, int, int, short);
int VGA_read_pixel(int, int) ;
int video_in_read_pixel(int, int);
void draw_delay(void) ;
int d2b(float num);
double b2d(int num);
double normpdf(int, int);
double Kull(int, int);
double dist_fn(int,int, int, int);

```

```
void maximum(double *a,double *max,int *max_index, int a_num);
void minimum(double *a,double *min, int *min_index, int a_num);
void nms(double *r);
void getcord(double *r);
```

```
int test_data[2][101];
// the light weight buss base
void *h2p_lw_virtual_base;
```

```
// RAM FPGA command buffer
volatile unsigned int * sram_ptr = NULL ;
void *sram_virtual_base;
```

```
// pixel buffer
volatile unsigned int * vga_pixel_ptr = NULL ;
void *vga_pixel_virtual_base;
```

```
// character buffer
volatile unsigned int * vga_char_ptr = NULL ;
void *vga_char_virtual_base;
```

```
// /dev/mem file id
int fd;
```

```
// pixel macro
// !!!PACKED VGA MEMORY!!!
#define VGA_PIXEL(x,y,color) do{\
    char *pixel_ptr ;\
    pixel_ptr = (char *)vga_pixel_ptr + ((y)*640) + (x) ;\
    *(char *)pixel_ptr = (color);\
} while(0)
```

```
// measure time
struct timeval t1, t2;
double elapsedTime;
struct timespec delay_time ;
```

```
int main(int argc, char *argv[])
{
    delay_time.tv_nsec = 10 ;
    delay_time.tv_sec = 0 ;
```

```

// Declare volatile pointers to I/O registers (volatile // means that IO load and store
instructions will be used // to access these pointer locations,
// instead of regular memory loads and stores)

// === need to mmap: =====
// FPGA_CHAR_BASE
// FPGA_ONCHIP_BASE
// HW_REGS_BASE

// === get FPGA addresses =====
// Open /dev/mem
if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
    printf( "ERROR: could not open \"/dev/mem\"...\n" );
    return( 1 );
}

// get virtual addr that maps to physical
// for light weight bus
h2p_lw_virtual_base = mmap( NULL, HW_REGS_SPAN, ( PROT_READ |
PROT_WRITE ), MAP_SHARED, fd, HW_REGS_BASE );
if( h2p_lw_virtual_base == MAP_FAILED ) {
    printf( "ERROR: mmap1() failed...\n" );
    close( fd );
    return(1);
}

// === get VGA char addr =====
// get virtual addr that maps to physical
vga_char_virtual_base = mmap( NULL, FPGA_CHAR_SPAN, ( PROT_READ |
PROT_WRITE ), MAP_SHARED, fd, FPGA_CHAR_BASE );
if( vga_char_virtual_base == MAP_FAILED ) {
    printf( "ERROR: mmap2() failed...\n" );
    close( fd );
    return(1);
}

// Get the address that maps to the character
vga_char_ptr =(unsigned int *) (vga_char_virtual_base);

// === get VGA pixel addr =====
// get virtual addr that maps to physical
// SDRAM
vga_pixel_virtual_base = mmap( NULL, FPGA_ONCHIP_SPAN, ( PROT_READ |
PROT_WRITE ), MAP_SHARED, fd, SDRAM_BASE); //SDRAM_BASE

```

```

    if( vga_pixel_virtual_base == MAP_FAILED ) {
        printf( "ERROR: mmap3() failed...\n" );
        close( fd );
        return(1);
    }
// Get the address that maps to the FPGA pixel buffer
vga_pixel_ptr =(unsigned int *)(vga_pixel_virtual_base);

// === get RAM FPGA parameter addr =====
sram_virtual_base = mmap( NULL, FPGA_ONCHIP_SPAN, ( PROT_READ |
PROT_WRITE ), MAP_SHARED, fd, FPGA_ONCHIP_BASE); //fp

if( sram_virtual_base == MAP_FAILED ) {
    printf( "ERROR: mmap3() failed...\n" );
    close( fd );
    return(1);
}
// Get the address that maps to the RAM buffer
sram_ptr =(unsigned int *)(sram_virtual_base);

// =====

/* create a message to be displayed on the VGA
and LCD displays */
/*
char text_top_row[40] = "DE1-SoC ARM/FPGA\0";
char text_bottom_row[40] = "Cornell ece5760\0";
*/
char num_string[20], time_string[50] ;

// a pixel from the video
int pixel_color;
// video input index
int i,j,k,l, count;

// clear the screen//0x9f
VGA_box (0, 0, 639, 479, 0x9f);
// clear the text
VGA_text_clear();

VGA_box(50, 420, 250, 470, 0x22); //start image

```


0155328027063277,-0.0107678509149574,-0.00988813713669263,-0.020453091196243,-0.04
22487677402954,-0.030835309318166,-0.0308057269012592,-0.0409124281748,-0.03995063
94444748,-0.039950606113565,-0.0405661439074018,-0.0345955691031752,-0.02302995594
21364,-0.033202984685275,-0.0337259839463448,-0.0173129988416091,0,0,0.20637286913
7841,0.405222803493719,0.399267647552195,0.27434204169746,0.423953275234977,0.486
392118396729,0.479914635821226,0.479915253521275,0.479915253521274,0.47991525352
1275,0.479915253521274,0.479915253521275,0.480678990220482,0.485458163153898,0.18
8454219303042,0.131381493777055,0.120663334625752,0.246878783215968,0.5073787307
10587,0.371037474151503,0.370573281581972,0.491409069464269,0.479915704461074,0.4
79915266170531,0.48726908120851,0.415945933509185,0.277711178265509,0.3992438082
42238,0.405624699491179,0.208097188888978,0,0,0.430588106793699,0.841442136575585,
0.827013021095587,0.567029775606461,0.879955960203802,1.01025801985091,0.99685163
5676047,0.996848230494828,0.9968482593067,0.9968482593067,0.996848259306699,0.996
848267700606,0.998497238823286,1.00815284917814,0.384910726633034,0.266862966350
954,0.247601311616156,0.511686443896295,1.05406179952184,0.769170665596574,0.7697
98597670489,1.02084788006652,0.996845558975659,0.996846905440005,1.0121882042529
6,0.863292339899213,0.574941888514922,0.828659493163887,0.841662817171691,0.43225
027108605,0,0,0.423754499316632,0.826860166419658,0.811747234292179,0.55638822268
3171,0.872773537830576,1.00377155139915,0.990605451172865,0.990529457030805,0.990
534146125869,0.990534098068068,0.990534098068068,0.990533954548466,0.99216336859
4975,1.00250679989325,0.376312190204504,0.2504278774832,0.234951655802511,0.50402
5992173302,1.04102105325253,0.758913671811481,0.764974038852552,1.01491530920394,
0.991074670159718,0.99102996241082,1.00631672881463,0.858826606762383,0.553428827
722653,0.795138691616781,0.807931631047485,0.414943212261097,0,0,0.28970175152413
1,0.562174421202065,0.552786481699105,0.37813044671,0.52167495692765,0.6009447941
92199,0.593562408574791,0.594224707780068,0.594149763831272,0.594151147565139,0.5
94151147565138,0.594152166111547,0.595063796236493,0.598631902288405,0.259378657
383592,0.243299125470245,0.225995774720389,0.341359062191665,0.665409235922101,0.
491931713362048,0.455375491232531,0.601705546048028,0.586906383164989,0.58738957
3215676,0.596279136941947,0.499148573820994,0.532320951657332,0.799059985246079,0
.807887369534212,0.414959279447899,0,0,0.45175804340143,0.88181777465559,0.8772838
55350579,0.536399215164234,0.177099994200124,0.224814054278281,0.222992134326948,
0.223291650966442,0.223264455935857,0.223264232540828,0.223264232540828,0.2232670
16028955,0.223572422849673,0.19788620194436,0.352604339774537,0.897023137640664,0
.831072529338655,0.474912372960927,0.724109605150142,0.553161743853867,0.18560015
9262347,0.233634233808387,0.229248756728145,0.22982154121138,0.232730528526108,0.
183844046709216,0.524467305074868,0.822337638023165,0.824612560305858,0.42288325
5035428,0,0,0.513859934495724,1.00391557598323,0.999893335632712,0.61308245167225
6,0.227673548221885,0.284307392596402,0.281428077365787,0.281397431846158,0.28140
0092218096,0.281400124825692,0.281400124825691,0.281399878897037,0.2818609871438
75,0.253946093063679,0.399349835051433,0.99310684499035,0.931808171561685,0.42197
2853778536,0.563642011360727,0.435622298164112,0.200518413875088,0.2536303306419
86,0.24715510584942,0.247148764188059,0.250875386732865,0.205749450142466,0.37285
9990668779,0.57069102230094,0.571555398649198,0.294975549182654,0,0,0.50751940256

8242,0.991334023453782,0.987432508592354,0.604703028441875,0.225102468032773,0.28
1257566670877,0.27843137254902,0.27843137254902,0.27843137254902,0.2784313725490
2,0.27843137254902,0.27843137254902,0.278871215830675,0.25144814414415,0.39497436
2977688,0.980890396600076,0.914611021042819,0.480164305093978,0.704809089306745,0
.531312013083569,0.661197205644065,0.881084459205777,0.862730017060967,0.86299647
9476928,0.873442383485835,0.739475237268445,0.582886721290792,0.864541229641237,0
.876288963802601,0.448761894362221,0,0,0.50751289371177,0.991332081676124,0.987416
064345768,0.604679652299098,0.225108067485299,0.281257222606144,0.27843137254902,
0.27843137254902,0.27843137254902,0.27843137254902,0.27843137254902,0.27843137254
902,0.278835747884896,0.251779230230826,0.394684411018283,0.980986945800216,0.909
348492800351,0.528089631993661,0.815041757577953,0.611564370894951,0.74531840721
5628,0.999566108667345,0.978716334618902,0.978776657780856,0.992442601145143,0.83
4445841572165,0.67006057648976,0.99709085349441,1.01038605141408,0.5170093285922
85,0,0,0.50751289371177,0.99133207825594,0.987416109823136,0.604684450822056,0.225
1080846453,0.281257215184855,0.27843137254902,0.278431372549019,0.27843137254901
9,0.27843137254902,0.278431372549019,0.278431372549019,0.278837861112033,0.251759
699781563,0.394719004878089,0.980997897714153,0.909653762593472,0.52307638862747
8,0.803985183305319,0.604651771662181,0.737101103926937,0.98069817074495,0.963807
447581865,0.964905533966874,0.979081821024568,0.822993801576867,0.66232840600666
2,0.985253434568328,0.999280105279637,0.513434221658319,0,0,0.50751289371177,0.991
33207825594,0.987416109823136,0.604684450822056,0.225108084645301,0.281257215184
855,0.27843137254902,0.27843137254902,0.27843137254902,0.27843137254902,0.2784313
7254902,0.27843137254902,0.2788378787983,0.251759528272461,0.394718549445729,0.98
0996352047076,0.909670109843301,0.523102310807744,0.803986204540531,0.6047062765
1783,0.737388590825335,0.980503185369373,0.9637644372649,0.964916603467077,0.9790
06084529499,0.822770756780195,0.664867050362549,0.991412702649881,1.003903412997
99,0.513627388356957,0,0,0.50751289371177,0.99133207825594,0.987416109823136,0.60
4684450822056,0.225108084645301,0.281257215184855,0.27843137254902,0.27843137254
902,0.27843137254902,0.27843137254902,0.27843137254902,0.27843137254902,0.2788378
787983,0.251759528272461,0.39471854944573,0.980996352047077,0.909670109843301,0.5
23102297199975,0.803987424667853,0.604695695787779,0.737354406172966,0.980546331
315124,0.963772700612403,0.964915634665271,0.979006014582781,0.822779585202226,0.
664791802907908,0.991192296105567,1.00371507631532,0.513590545083044,0,0,0.507512
89371177,0.99133207825594,0.987416109823136,0.604684450822056,0.2251080846453,0.2
81257215184855,0.27843137254902,0.278431372549019,0.278431372549019,0.2784313725
4902,0.278431372549019,0.278431372549019,0.2788378787983,0.251759528272461,0.3947
18549445729,0.980996352047076,0.909670224778515,0.523101799005014,0.803988487987
962,0.604695678293247,0.737354406172965,0.980546331315123,0.963772700612402,0.964
915634665269,0.979006038854322,0.822779866676874,0.664786584761183,0.99118263441
4145,1.00370858232839,0.513590619039938,0,0,0.507500046806382,0.991313702517716,0.
987398812000491,0.604666527420025,0.225080169072433,0.281307823070981,0.27848424
1509029,0.278483241386956,0.278483280118424,0.278483280118424,0.278483280118424,0
.278482859936717,0.27889352647734,0.251852780754217,0.394686182803972,0.980961232
69763,0.909661327330889,0.523068767332293,0.803965996852269,0.604696687671223,0.7

37354407626692,0.980546331315124,0.963772700612403,0.96491563466527,0.9790060388
54323,0.822779866676874,0.664786584761183,0.991182634414145,1.00370858232839,0.51
3590619039939,0,0,0.525419389412856,1.0266186500657,1.02272973680899,0.6245291651
52125,0.217182655197599,0.272906271061348,0.270145263822829,0.270156075400974,0.2
70155806987358,0.270155806987359,0.270155806987358,0.270159445105303,0.270545189
807304,0.242045127057818,0.392255567229623,1.00303392489631,0.93010025665281,0.52
6887339215776,0.803598490492802,0.604687098782824,0.73735375161329,0.98054634251
3688,0.963772700612403,0.964915634665271,0.979006038854324,0.822779866676875,0.66
4786584761183,0.991182634414146,1.00370858232839,0.513590619039939,0,0,0.27322297
3294858,0.528982441485518,0.525174489213477,0.338744261320096,0.247528224779984,0
.288396105320827,0.285413293228262,0.285456887211587,0.285453389180438,0.28545338
9180438,0.285453389180437,0.285463288544947,0.285596186649202,0.271327864141551,0
.336695512245342,0.61434489501288,0.561830624980302,0.466256844137129,0.811791620
313833,0.604688992038147,0.737355370220328,0.980544427777887,0.963772793088158,0.
964915634665269,0.97900601356796,0.822780129792977,0.664786978790379,0.991182638
792296,1.00370858115318,0.513590619039938,0,0,0.129778377748054,0.245698750998596,
0.233192211469511,0.215751905915765,0.825868121809791,0.927601736817575,0.9142250
58192199,0.914128987190671,0.914136568917064,0.914136557874695,0.914136557874694,
0.914116550867327,0.914172427208,0.916777434933018,0.912480278026759,0.8933426068
61783,0.843317837356253,0.510871932392657,0.805356184021717,0.604431024961538,0.7
37305960797528,0.980554315768581,0.963775988653139,0.964915232770812,0.979000792
355303,0.822825971419902,0.664790074606742,0.991189467579007,1.00373266997251,0.5
13596236972633,0,0,0.15908980239482,0.304110152604227,0.290951160465739,0.2511651
46918468,0.869312813746074,0.979564204856165,0.965498144557398,0.963903086167789,
0.964102508124838,0.964095739819977,0.964095739819977,0.964097084364122,0.9640506
68507824,0.964315338332159,0.965716918956029,0.971667467026478,0.921128984512725,
0.53383437303698,0.824900580100307,0.621733692667567,0.762189521595659,1.01169206
963829,0.994166137576523,0.995412552352782,1.01004970515892,0.847668434312417,0.6
84556761362074,1.02231192589394,1.03499004008855,0.52945112773898,0,0,0.123209561
026011,0.236000839733805,0.22276888463188,0.209896123335674,0.8687266308135,0.976
596076047655,0.959703370258971,0.957149451872171,0.95645308623429,0.956533678961
624,0.956533678961623,0.956533678961625,0.956537490206916,0.956505063195568,0.956
35326336158,0.961958739438429,0.916985387434085,0.431656298827122,0.589446905883
937,0.450266067980484,0.523453987134176,0.69235017629414,0.677568854555586,0.6789
59381093039,0.688921795855783,0.582091300541969,0.473582094115941,0.696261417658
414,0.704862286025239,0.362882409564958,0,0,0.124737050966125,0.236617762808385,0.
224866016216218,0.211055395127793,0.868987111985521,0.976913254559979,0.96049524
9557384,0.961200181344001,0.961307128420784,0.961293396774114,0.961293396774113,0
.961293396774114,0.961293396774114,0.961293396774114,0.961292964347212,0.96699241
4685368,0.911145133443685,0.52739441861556,0.829121989613121,0.624322009091357,0.
170197716512619,0.219575351320822,0.215818669478785,0.215824155987405,0.21550009
148605,0.219748610389218,0.225189073591914,0.218772116099835,0.228470047619787,0.
12070383278708,0,0,0.157627334493089,0.300427042638497,0.2883198385407,0.24618489
4607019,0.863992316988069,0.971722927764957,0.956516943298211,0.963441353036395,0


```

        *pio_rst_ptr=1;
        *(sram_ptr) = 1;
        gettimeofday(&t1, NULL);
        printf("Reset is detected\n");
    }
    // start the timer
    draw_delay();
    *pio_rst_ptr=0;
    printf("reset is now 0\n");
    /*(sram_ptr) = 1;
    printf("hps is ready\n");
    while (*(sram_ptr)!=1);
    gettimeofday(&t2, NULL);
    elapsedTime = (t2.tv_sec - t1.tv_sec) * 1000000.0; // sec to us
    elapsedTime += (t2.tv_usec - t1.tv_usec) ; // us to
    //sprintf(num_string, "# = %d ", total_count);
    sprintf(time_string, "T=%8.0f uSec ", elapsedTime);
    printf("T=%8.0f uSec\n", elapsedTime);
    VGA_text (10, 3, num_string);
    VGA_text (10, 4, time_string);

    double result[900];
    for (i = 1; i <= 900; i++) {
        result[i-1] = b2d(*(sram_ptr + i)) * 16.0;
    }
    result[899] = 0;// force to correct
    //covert decimal to fixed point
    /*for(i=0; i < 900; i++){
        printf("%f\n", result[i]);
    }*/

    nms(result);

    for (i = 1; i <= 900; i++) {
        if(result[i-1] > 0){
            VGA_disc(12*((i-1)%30-1)+152,12*((i-1)/30-1)+32,5,0xe0);
        }
        //printf("The r value of %d is: %f\n", i, result[i-1]);
    }
    getcord(result);
    //
    //printf("Thes number of corners is %d\n", test_data[0][0]);
    /*for (i = 0; i < test_data[0][0]; i++) {
        printf("[%d, %d]\n", test_data[0][i+1], test_data[1][i+1]);
    }

```



```

int train_num = 45;
double score[train_num][3][26]; // score[num_database][three different stastics][26 letter];

for(l=0; l < train_num; l++){
    for(k=0; k < 26; k++) {

        ///compare data
        //first step: compare both data length
        int data_length = (train_data[l][k][0][0] > test_data[0][0]) ? test_data[0][0] :
train_data[l][k][0][0];
        //printf("length: %d \n", data_length);
        //second: rearrange array order based on the distance
        int order[21]; // the val at index i is already been shifted, i.e the
        int check[100];

        for (i = 0; i < 100; i++) {
            check[i] = 0;
        }
        for (i = 0; i < data_length; i++) {
            float min_dist = 42;
            int best_fit = 22;
            for (j = 0; j < test_data[0][0]; j++) {
                float dist = dist_fn(test_data[0][j+1], train_data[l][k][0][i+1],
test_data[1][j+1], train_data[l][k][1][i+1]);
                //printf("the dist of train data%d --> test data%d: %f\n",i, j,
dist);

                if (check[j] == 0) {
                    if (dist < min_dist) {
                        min_dist = dist;
                        best_fit = j;
                        //printf("update min_dist for %d --> %d, now
is %f\n",i, j, min_dist);
                    }
                }
            }
            check[best_fit] = 1;
            order[i] = best_fit + 1;
            //printf("%d\n", order[i]);
        }

        //Third do the normal pdf and Kull
        //1. sum(normalpdf(x) .* pdf(y)

```

```

        double Pdf_dot = 0.0;
        for (i = 0; i < data_length; i++){
            Pdf_dot = Pdf_dot + normpdf(test_data[0][order[i]],
train_data[l][k][0][i+1]) * normpdf(test_data[1][order[i]], train_data[l][k][1][i+1]);
        }
        //2. normalpdf: sum(x) * sum(y)
        score[l][0][k] = Pdf_dot;
        double P_x = 0.0;
        double P_y = 0.0;
        for (i = 0; i < data_length; i++){
            P_x = P_x + normpdf(test_data[0][order[i]], train_data[l][k][0][i+1]);
            P_y = P_y + normpdf(test_data[1][order[i]], train_data[l][k][1][i+1]);
        }
        score[l][1][k] = P_x * P_y;
        //3. Kull(x) * Kull (y)
        double K_x = 0.0;
        double K_y = 0.0;
        for (i = 0; i < data_length; i++){
            K_x = K_x + Kull(test_data[0][order[i]], train_data[l][k][0][i+1]);
            K_y = K_y + Kull(test_data[1][order[i]], train_data[l][k][1][i+1]);
            //printf("Kx: %d, Ky: %d Kx: %d, Ky: %d\n", test_data[0][order[i]],
test_data[1][order[i]], train_data[l][k][0][i+1], train_data[l][k][1][i+1]);
        }
        if(K_x * K_y < 0){
            score[l][2][k] = -1.0 * K_x * K_y;
        }
        else{
            score[l][2][k] = K_x * K_y;
        }
        //printf("idx:%d, %d %d Pdf1: %f, Pdf2: %f K: %f \n", l, k, data_length,
score[l][0][k], score[l][1][k],score[l][2][k]);
        //third compare result
    }
    //printf("next: \n");
} //end compare to train database

double list[3][26] =
{{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};
double max = 0;
int max_index = 0;
double min = 999;
int min_index = 0;
//start compare

```

```

for(l = 0; l < train_num; l++){
//score1
    max = 0;
    max_index = 0;

    maximum(score[l][0], &max, &max_index, 26);
    list[0][max_index] = list[0][max_index] + 1.0;
    //printf("idx: %d \n", max_index);
//score2
    max = 0;
    max_index = 0;
    maximum(score[l][1], &max, &max_index, 26);
    list[1][max_index] = list[1][max_index] + 1.0;
//score3
    min = 999;
    min_index = 0;
    minimum(score[l][2], &min, &min_index, 26);
    list[2][min_index] = list[2][min_index] + 1.0;
}
/*for(j = 0; j < 3; j++){
for(i=0; i<26; i++){
    printf("%f ", list[j][i]);
}
printf("\n");
}*/

//find the correspond letter
char Text_arr[26]
= {'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
int text_idx1,text_idx2,text_idx3;
double val1,val2,val3;

for(l = 0; l < 3; l++){
    max = 0;
    max_index = 0;
    maximum(list[l], &max, &max_index, 26);

    if(l == 0){
        text_idx1 = max_index;
        val1 = max;
    }
    else if(l == 1){
        text_idx2 = max_index;

```

```

        val2 = max;
    }
    else{
        text_idx3 = max_index;
        val3 = max;
    }
}
printf("first:%d, %f second:%d, %f third:%d, %f\n",
text_idx1,val1,text_idx2,val2,text_idx3,val3);
double tmp_val[3]={val1,val2,val3};
int tmp_idx[3]={text_idx1,text_idx2,text_idx3};
int tmp_buff[3] = {3,3,3};
int tmp_count = 0;
char final_ans;
//determine the text
if(text_idx1 == text_idx2 || text_idx1 == text_idx3){
    final_ans = Text_arr[text_idx1];
}
else if(text_idx2 == text_idx3){
    final_ans = Text_arr[text_idx2];
}
else{
    max = 0;
    max_index = 0;
    maximum(tmp_val, &max, &max_index, 3);
    for(i = 0; i < 3; i++){ //check if there is two factors have same weight
        if(tmp_val[i] == max){
            tmp_buff[i] = i;
            tmp_count = tmp_count + 1;
        }
    }
    if(tmp_count == 2){
        if(tmp_buff[0] == 0 && tmp_buff[1] == 2){
            final_ans = Text_arr[text_idx1];
        }
        else{
            final_ans = Text_arr[text_idx2];
        }
    }
    else{
        final_ans = Text_arr[tmp_idx[max_index]];
    }
}
}

```

```

for(i=0;i <26; i++){
    if(Text_arr[i] == user_char){
        user_input = i;
    }
}
char InputInfo[40] ="The input image is .\0";
char Outputmsg[40] ="The letter should be .\0";
InputInfo[19] = Text_arr[user_input];
Outputmsg[21] = final_ans;

```

```

VGA_box(50, 420, 250, 470, 0x22); //finish text image

```

```

int finish_msg[336] =
{0,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,0,0,0,1,0,0,0,1,1,1,1,0,0,0,1,1,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,
0,0,1,0,0,0,0,1,1,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,
0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,1,0,0,0,0,1,0,
0,0,0,0,1,1,1,0,0,0,1,1,1,1,1,0,0,1,1,1,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,
0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,0,0,1,
0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,1,1,1,
1,1,0,0,1,0,0,0,1,0,0,1,1,1,1,1,0,0,0,1,1,1,0,0,0,1,0,0,0,1,0,0};
    for (j=2; j < 338; j++){
        if(finish_msg[j-2] == 1){
            VGA_box(3*((j-2)%42-1) + 90, 3*((j-2)/42-1)+435, 3*((j-2)%42-1) +
93, 3*((j-2)/42-1)+438, 0xff);
        }
    }
short text_color;
if(Text_arr[user_input] == final_ans){
    text_color = 0x28;
    VGA_box(400, 420, 600, 470, text_color);
    int bingo_msg[280]
={0,1,1,1,1,0,0,0,1,1,1,1,1,0,0,1,0,0,0,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0
,1,1,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,0,0,1,0,0,1,0,0,0,0,0,1,0
,0,1,0,0,1,1,1,1,0,0,0,0,0,1,0,0,0,0,1,0,1,0,1,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,1,0
,0,0,1,0,1,0,1,0,0,1,0,1,1,1,0,0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0,0,1,0,0,0,1,0
,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0,1,0,0,0,1,0,0,1,0,0,0,1,0,0,1,1,1,1,0,0,0,1
,1,1,1,1,0,0,1,0,0,0,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,0};
    int smile[225] =
{2,2,2,2,2,1,1,1,1,1,2,2,2,2,2,2,2,2,1,1,0,0,0,0,0,1,1,2,2,2,2,2,1,0,0,0,0,0,0,0,0,0,1,2,2,2,1,0,0,0,
0,0,0,0,0,0,0,0,1,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,1,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,2,2,1,1,1,1,2,2,
1,1,0,1,1,0,1,2,1,1,1,0,1,2,1,1,1,0,1,1,0,0,1,1,1,0,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,1,

```

```

0,0,0,0,0,0,0,0,1,0,0,1,2,2,1,0,0,0,1,1,1,1,1,0,0,0,1,2,2,2,1,0,0,0,0,0,0,0,0,0,1,2,2,2,2,2,1,1,0,0,0,
0,0,1,1,2,2,2,2,2,2,2,1,1,1,1,1,2,2,2,2,2};
    short smile_color;
    for (i=2; i < 282; i++){
        if(bingo_msg[i-2] == 1){
            VGA_box(3*((i-2)%35-1) + 450, 3*((i-2)/35-1)+435, 3*((i-2)%35-1)
+ 453, 3*((i-2)/35-1)+438, 0xff);
        }
    }

    for (i=2; i < 227; i++){
        if(smile[i-2] == 1){
            smile_color = 0x00;
        }
        else if(smile[i-2] == 2){
            smile_color = 0x9f;
        }
        else{
            smile_color = 0xa4;
        }
        VGA_box(3*((i-2)%15-1) + 315, 3*((i-2)/15-1)+423, 3*((i-2)%15-1) + 318,
3*((i-2)/15-1)+426, smile_color);
    }

}
else{
    text_color = 0x60;
    VGA_box(400, 420, 600, 470, text_color);

    int fail_msg[224] =
{0,1,1,1,1,1,0,0,0,1,1,1,0,0,0,1,1,1,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,
1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,
1,0,0,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,
1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,
0,1,0,0,0,1,0,0,1,1,1,1,1,0,0,1,1,1,1,1,0};
    int sad[225] =
{2,2,2,2,2,1,1,1,1,1,2,2,2,2,2,2,2,2,1,1,0,0,0,0,0,1,1,2,2,2,2,2,1,0,0,0,0,0,0,0,0,0,1,2,2,2,1,0,1,1,
1,0,0,0,1,1,1,0,1,2,2,1,0,0,0,0,0,0,0,0,0,0,0,1,2,1,0,0,1,0,1,0,0,0,1,0,1,0,0,1,1,0,0,0,1,0,0,0,0,1,
0,0,0,1,1,0,0,1,0,1,0,0,0,1,0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,1,
0,0,0,0,1,1,1,0,0,0,0,1,2,2,1,0,0,0,1,0,0,0,1,0,0,0,1,2,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,1,2,2,2,2,2,1,1,0,0,0,
0,0,1,1,2,2,2,2,2,2,2,2,1,1,1,1,1,2,2,2,2,2};

```

```

        short sad_color;
        for (i=2; i < 226; i++){
            if(fail_msg[i-2] == 1){
                VGA_box(3*((i-2)%28-1) + 470, 3*((i-2)/28-1)+435, 3*((i-2)%28-1)
+ 473, 3*((i-2)/28-1)+438, 0xff);
            }
        }
        for (i=2; i < 227; i++){
            if(sad[i-2] == 1){
                sad_color = 0x00;
            }
            else if(sad[i-2] == 2){
                sad_color = 0x9f;
            }
            else{
                sad_color = 0xc4;
            }
            VGA_box(3*((i-2)%15-1) + 315, 3*((i-2)/15-1)+423, 3*((i-2)%15-1) + 318,
3*((i-2)/15-1)+426, sad_color);
        }
    }
    //VGA_text(70,50, "HELLO");

    printf("The letter should be %c. \n", Outputmsg[21]);

```

```

} // end main

```

```

//normal distribution

```

```

double normpdf(int x, int mu){

```

```

    float sigma = 2.0;

```

```

    double y = (1.0 / (sigma * sqrt(2*M_PI))) * exp(-(x - mu)*(x - mu) / (2.0 * sigma * sigma));

```

```

    return y;

```

```

}

```

```

//Kullback-Leibler

```

```

double Kull(int x, int y){

```

```

    int array_size = 30;

```

```

    double ans = (1.0 * x / array_size) * log( x * 1.0 / y );

```

```

    return ans;

```

```

}

```



```
//distance function
double dist_fn(int test_x, int train_x, int test_y, int train_y){
    double ans = sqrt(pow((test_x - train_x),2) + pow((test_y - train_y),2));
    return ans;
}
```

```
/******
 * Subroutine to read a pixel from the video input
*****/
// int video_in_read_pixel(int x, int y){
    // char *pixel_ptr ;
    // pixel_ptr = (char *)video_in_ptr + ((y)<<9) + (x) ;
    // return *pixel_ptr ;
// }
```

```
/******
 * Subroutine to read a pixel from the VGA monitor
*****/
int VGA_read_pixel(int x, int y){
    char *pixel_ptr ;
    pixel_ptr = (char *)vga_pixel_ptr + ((y)*640) + (x) ;
    return *pixel_ptr ;
}
```

```
/******
 * Subroutine to send a string of text to the VGA monitor
*****/
void VGA_text(int x, int y, char * text_ptr)
{
    volatile char * character_buffer = (char *) vga_char_ptr ; // VGA character buffer
    int offset;
    /* assume that the text string fits on one line */
    offset = (y << 7) + x;
    while ( *(text_ptr) )
    {
        // write to the character buffer
        *(character_buffer + offset) = *(text_ptr);
        ++text_ptr;
        ++offset;
    }
}
```

```

}

/*****
 * Subroutine to clear text to the VGA monitor
 *****/
void VGA_text_clear()
{
    volatile char * character_buffer = (char *) vga_char_ptr ;    // VGA character buffer
    int offset, x, y;
    for (x=0; x<79; x++){
        for (y=0; y<59; y++){
            /* assume that the text string fits on one line */
            offset = (y << 7) + x;
            // write to the character buffer
            *(character_buffer + offset) = ' ';
        }
    }
}

```

```

/*****
 * Draw a filled rectangle on the VGA monitor
 *****/
#define SWAP(X,Y) do{int temp=X; X=Y; Y=temp;}while(0)

void VGA_box(int x1, int y1, int x2, int y2, short pixel_color)
{
    char *pixel_ptr ;
    int row, col;

    /* check and fix box coordinates to be valid */
    if (x1>639) x1 = 639;
    if (y1>479) y1 = 479;
    if (x2>639) x2 = 639;
    if (y2>479) y2 = 479;
    if (x1<0) x1 = 0;
    if (y1<0) y1 = 0;
    if (x2<0) x2 = 0;
    if (y2<0) y2 = 0;
    if (x1>x2) SWAP(x1,x2);
    if (y1>y2) SWAP(y1,y2);
    for (row = y1; row <= y2; row++)
        for (col = x1; col <= x2; ++col)
        {
            //640x480

```

```

        VGA_PIXEL(col, row, pixel_color);
        //pixel_ptr = (char *)vga_pixel_ptr + (row<<10) + col ;
        // set pixel color
        /*(char *)pixel_ptr = pixel_color;
    }
}

/*****
* Draw a filled circle on the VGA monitor
*****/

void VGA_disc(int x, int y, int r, short pixel_color)
{
    char *pixel_ptr ;
    int row, col, rsqr, xc, yc;

    rsqr = r*r;

    for (yc = -r; yc <= r; yc++)
        for (xc = -r; xc <= r; xc++)
        {
            col = xc;
            row = yc;
            // add the r to make the edge smoother
            if(col*col+row*row <= rsqr+r){
                col += x; // add the center point
                row += y; // add the center point
                //check for valid 640x480
                if (col>639) col = 639;
                if (row>479) row = 479;
                if (col<0) col = 0;
                if (row<0) row = 0;
                VGA_PIXEL(col, row, pixel_color);
                //pixel_ptr = (char *)vga_pixel_ptr + (row<<10) + col ;
                // set pixel color
                //nanosleep(&delay_time, NULL);
                //draw_delay();
                /*(char *)pixel_ptr = pixel_color;
            }
        }
}

// =====

```

```

// === Draw a line
// =====
//plot a line
//at x1,y1 to x2,y2 with color
//Code is from David Rodgers,
//"Procedural Elements of Computer Graphics",1985
void VGA_line(int x1, int y1, int x2, int y2, short c) {
    int e;
    signed int dx,dy,j, temp;
    signed int s1,s2, xchange;
signed int x,y;
    char *pixel_ptr ;

    /* check and fix line coordinates to be valid */
    if (x1>639) x1 = 639;
    if (y1>479) y1 = 479;
    if (x2>639) x2 = 639;
    if (y2>479) y2 = 479;
    if (x1<0) x1 = 0;
    if (y1<0) y1 = 0;
    if (x2<0) x2 = 0;
    if (y2<0) y2 = 0;

    x = x1;
    y = y1;

    //take absolute value
    if (x2 < x1) {
        dx = x1 - x2;
        s1 = -1;
    }

    else if (x2 == x1) {
        dx = 0;
        s1 = 0;
    }

    else {
        dx = x2 - x1;
        s1 = 1;
    }

    if (y2 < y1) {
        dy = y1 - y2;

```

```

        s2 = -1;
    }

    else if (y2 == y1) {
        dy = 0;
        s2 = 0;
    }

    else {
        dy = y2 - y1;
        s2 = 1;
    }

    xchange = 0;

    if (dy>dx) {
        temp = dx;
        dx = dy;
        dy = temp;
        xchange = 1;
    }

    e = ((int)dy<<1) - dx;

    for (j=0; j<=dx; j++) {
        //video_pt(x,y,c); //640x480
        VGA_PIXEL(x, y, c);
        //pixel_ptr = (char *)vga_pixel_ptr + (y<<10)+ x;
        // set pixel color
        /*(char *)pixel_ptr = c;

        if (e>=0) {
            if (xchange==1) x = x + s1;
            else y = y + s2;
            e = e - ((int)dx<<1);
        }

        if (xchange==1) y = y + s2;
        else x = x + s1;

        e = e + ((int)dy<<1);
    }
}

```



```

// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10();
// NOP10(); NOP10(); NOP10(); NOP10(); //96
}

```

```

int d2b(float num) {
    float num_abs = (num > 0) ? num : num * (-1);
    int decimal = num_abs;
    float frac = num_abs - decimal;
    int base = decimal << 21;
    float remain = frac;
    int result = 0;
    int i;
    for (i = -1; i > -21; i--) {
        if (remain >= pow(2, i)) {
            remain -= pow(2, i);
            result = result | 0x1;
        }
        result = result << 1;
    }
    if (num < 0) {
        return ~(result + base) + 0x1;
    }
    return result + base;
}

```

```

double b2d(int num) {
    double result = 0;
    int decimal = num & 0x1fffff;
    int i, j;
    for (i = 21; i > 0; i--) {
        result += (decimal & 0x1) / pow(2, i);
        decimal = decimal >> 1;
    }
}

```



```

int fraction = num >> 21;
for (j = 0; j < 4; j++) {
    result += (j == 3) ? (-1 * (fraction & 0x1) * pow(2, j)) : ((fraction & 0x1) * pow(2, j));
    fraction = fraction >> 1;
}
return result;
}

```

```

void maximum(double *a, double *max, int *max_index, int a_num){
    int i;
    for (i=0; i<a_num; i++){
        //printf("array: %f\n", a[i]);
        if (a[i] > *max){
            *max = a[i];
            *max_index = i;
        }
    }
}

```

```

void minimum(double *a, double *min, int *min_index, int a_num){
    int i;
    for (i=0; i<a_num; i++){
        if (a[i] < *min){
            *min = a[i];
            *min_index = i;
        }
    }
}

```

```

void nms(double *r){
    int i, j, ii, jj;
    for (i=0; i<28; i++){
        for (j=0; j<28; j++){
            double max = 0;
            int idx = 0;
            for (ii=0; ii<3; ii++){
                for (jj=0; jj<3; jj++){
                    if (r[(i+ii)*30+j+jj] > max){
                        max = r[(i+ii)*30+j+jj];
                        idx = (i+ii)*30+j+jj;
                    }
                }
            }
        }
    }
    for (ii=0; ii<3; ii++){

```

