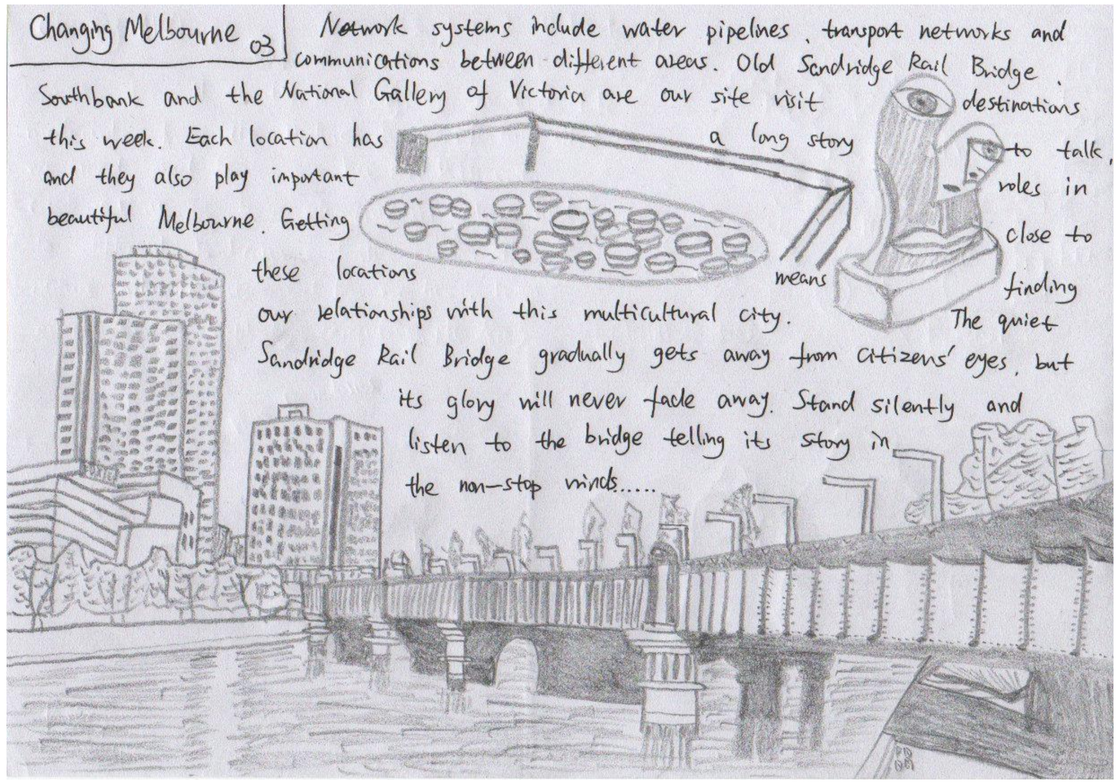
Week Three

Scanned journal:

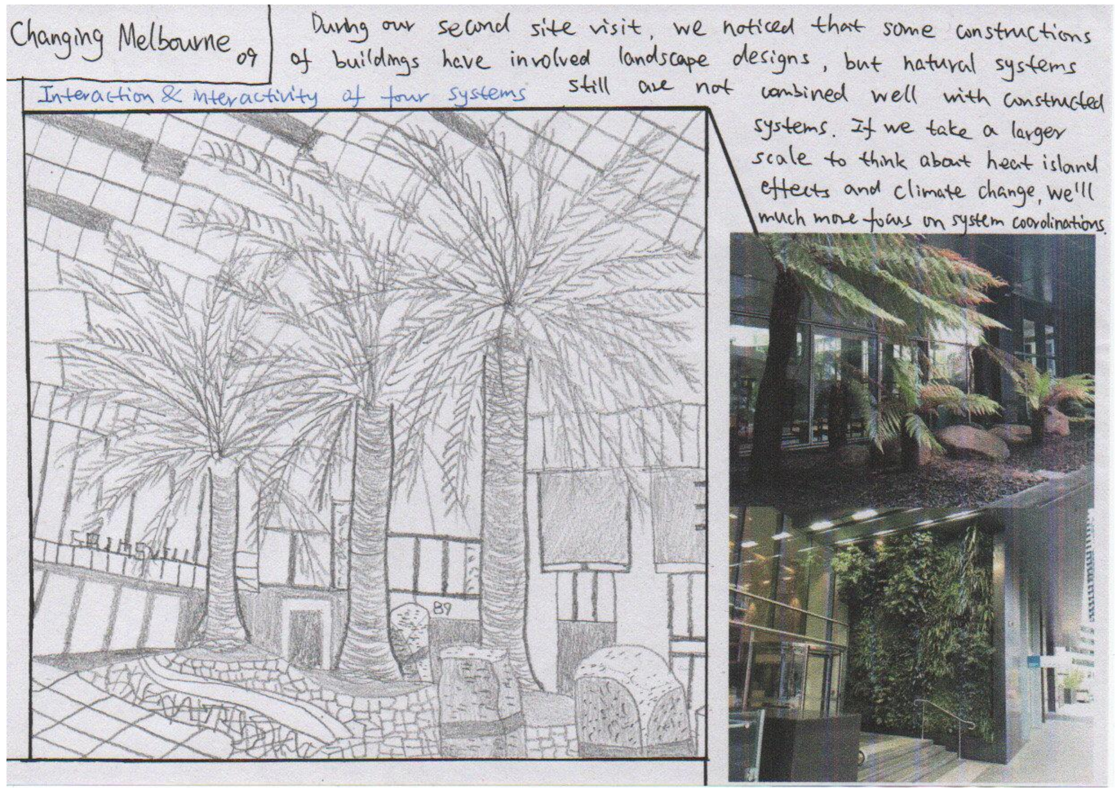


Reflections:

Through the site visit this week, I learn more about the city Melbourne and gradually fall in love with this city. Old Sandridge Rail Bridge represents 19th-century's industrial technology and is the most highly decorated railway bridge in the state *("Sandridge Bridge", 2016)*. Although in many people's eyes this bridge has lost its colors and declined, we still cannot deny the bridge has undertaken its responsibility all the time--it was an important railway line and now is a part of Flinders Walk. Time can change everything, but this bridge witnessed the process of constructing the city as well as its prosperity. Walking along the Yarra River, sculptures, bridges and buildings are telling me long stories about Melbourne and spirits of arts. The National Gallery of Victoria, belonging to the social system, is an example of the Melbourne's spirit. Different types of work show the culture diversities and cultural collisions, which make us closer to arts and our life much more colorful. Probably someday I would feel it is my honor to live in such a multicultural city.

Week Nine

Scanned journal:



Reflections:

Materials from natural environments are taken to construct buildings from thatched shacks to skyscrapers. Functions of buildings have changed a lot from living in villages to commercial uses in megacities. Some objects belong to multiple systems, such as bridges can be seen as a part of constructed systems and network systems. Therefore, combination of constructed systems and natural systems which give more functions to buildings can be a good approach to solve problems such as climate warming caused by Heat Island effect in cities. This is also a good way to make the most uses and potentials of current buildings. In additional, some architectures such as Ken Yeang think more ecological functions can be given to tall building rather than inhibiting unstoppable tall buildings because of urbanization ("Ken Yeang", 2016).

Two pages from Engineering Materials Assignment 1, 2016 Semester 2.
Figure 2.1~2.5 were drawn by hands.

Plan View and Elevation Views

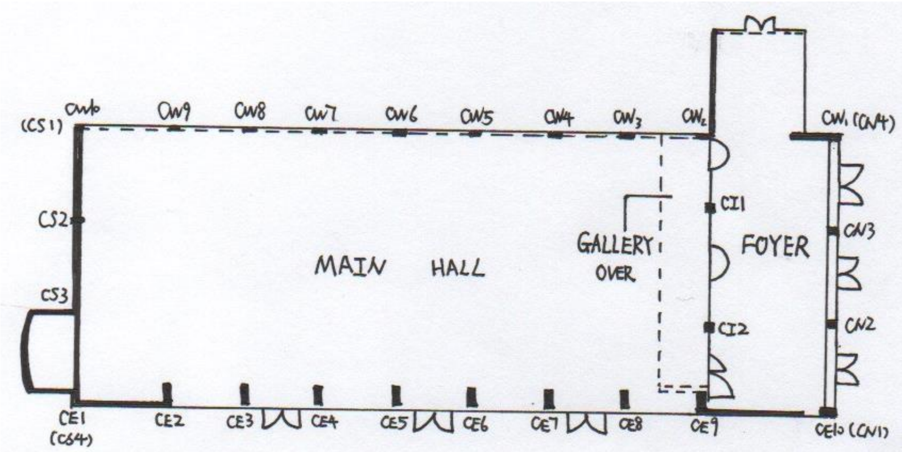
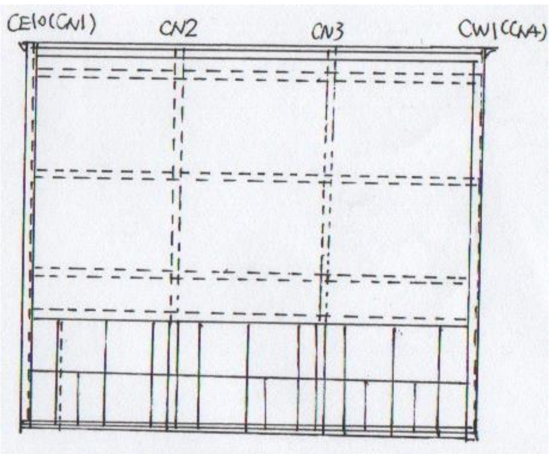


Figure 2.1 Plan View of Wilson Hall (Ground Floor)

Wilson Hall is known for its four different external facades: 1) The main entrance of the building is on the North Wall whose upper part are infilled brickworks as well as the glass doors placed between columns make up the below part; 2) The South Wall's textured brickworks include stone rosettes from the old building; 3) Four famous relief sculptures are on the West Wall which is constituted by ten columns and orange masonry wall; 4) The East Wall mainly is 36m(long)* 14.5m(high) heat absorbing glass wall with steel frames. Therefore, it is vital to show elevations of the building by sketches of four walls:



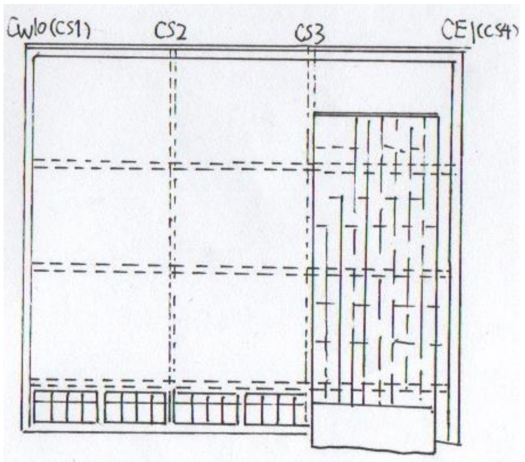


Figure 2.2 North Elevation View of Wilson Hall

Figure 2.3 South Elevation View of Wilson Hall

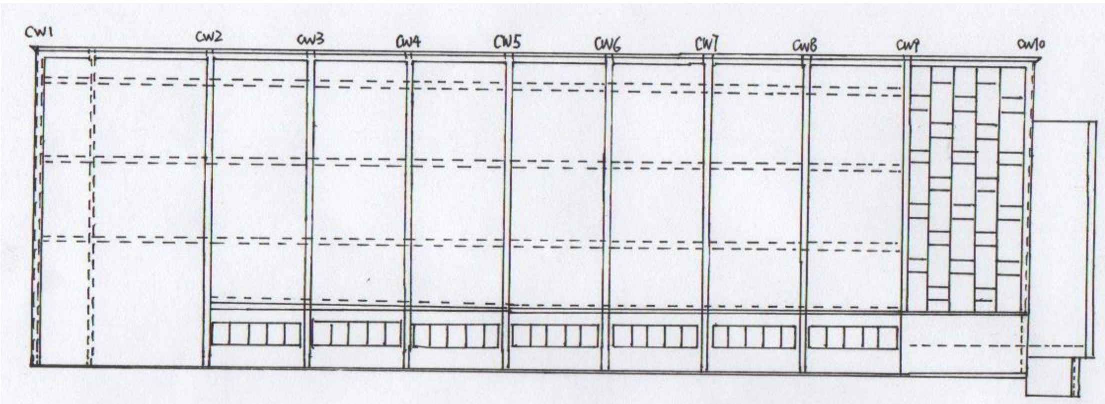


Figure 2.4 West Elevation View of Wilson Hall

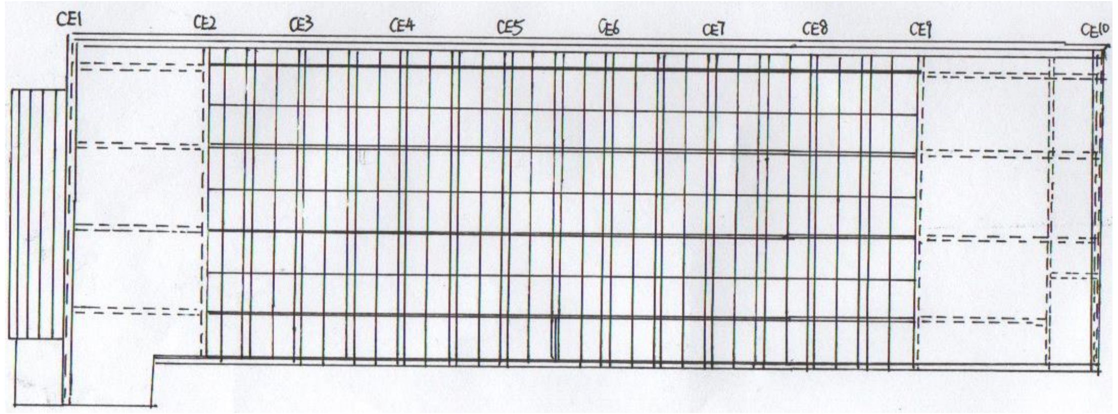


Figure 2.5 East Elevation View of Wilson Hall

Wall	Load Carrying Elements	
North Wall	Reinforced concrete column, Reinforced concrete beam, Bricks	
South Wall	Reinforced concrete column, Reinforced concrete beam, Bricks	
East Wall	External Glass Wall	Interior
	Steel frames, Glass pane	Ten Reinforced concrete column,
West Wall	Reinforced concrete column, Reinforced concrete beam, Bricks	

Table 2.1 Load Carrying Elements of Walls

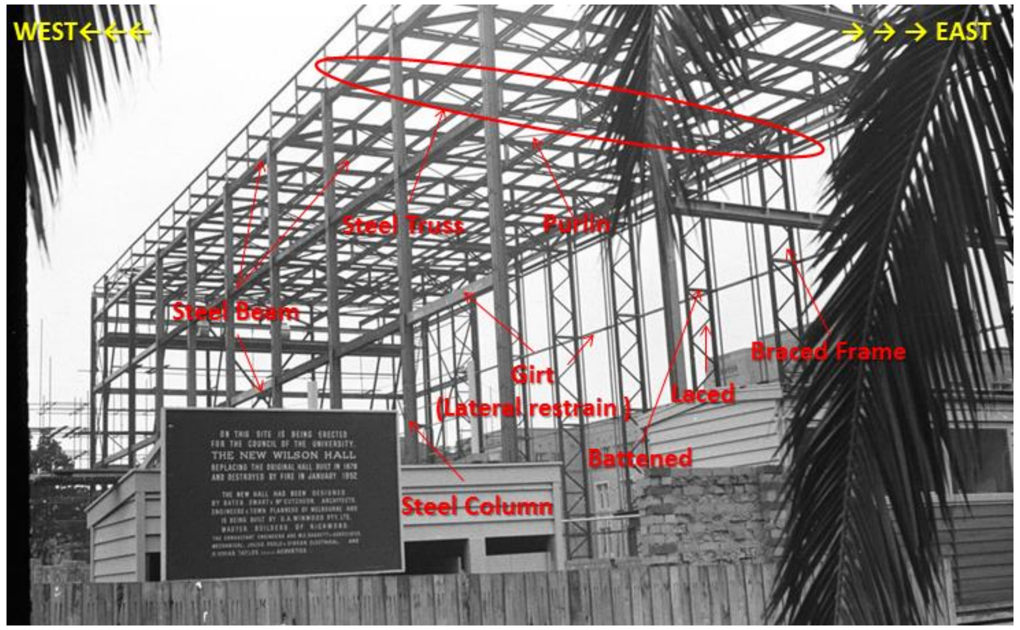


Figure 2.6 the Steel Structure of Wilson Hall

After constructing the foundation of the Wilson Hall, steel frames and steel trusses were then settled because better tension strength of the steel can alleviate the disadvantages of concrete and masonry when they are under tensions. Though steel has better tension strengthen, long steel load carrying elements will be buckled rather than broken because buckling strength is smaller than crushing strength. Therefore, steel beams and girts which provide lateral restrain are used to avoid material failure caused by buckling when steel columns are in structures. Combining with the better compression strength owned by concrete and masonry, the frames and structures of the building can show much better appearance under both tensions and compressions.

```c
/* ============================================================ */
/* This program is for COMP20005 Engineering Computation 2016S2 Assignment 2
 * Shaobin Zhao, ID:776298, shaobinz@student.unimelb.edu.au, October 2016 */
/* ============================================================ */
/* This program uses a function: int line_intersect(line_t l1, line_t l2);
 *
 * The function was adapted in 2012 from
 * http://local.wasp.uwa.edu.au/~pbourke/geometry/lineline2d/pdb.c
 * (no longer available at that URL in 2013)
 * and was written in the first instance by Paul Bourke
 *
 * Modified for use in a different assignment previously by Alistair Moffat by:
 *   . changing the argument type to two structs type line_t
 *   . making sure result is TRUE if an endpoint is on the other line
 *
 * Modified for use by Jianzhong Qi by:
 *   . testing whether the projections of the two line segments intersect first
 */

#include <stdio.h>
#include <math.h>


/*======================================================================*/
/* Define some constants and symbols */
#define MAX_ELE_NUM 99
#define ORIGIN_COOR_X 00.0
#define ORIGIN_COOR_Y 00.0
#define REGION_LEN 78
#define S3_TESTED_STR -50
#define STR_POS '+'
#define STR_NEG_TEN ' '
#define STR_NEG_TWE '2'
#define STR_NEG_THI ' '
#define STR_NEG_FOR '4'
#define STR_NEG_FIF ' '
#define STR_NEG_SMA_FIF '-'
#define TRUE 1
#define FALSE 0
#define EPS (1e-06)
#define ABS(x) (fabs(x))
#define MIN(a,b) (a<b ? a:b)
#define MAX(a,b) (a>b ? a:b)


/*======================================================================*/
/* Create structures of data types used in the program */
typedef struct {
   double x;
   double y;
} point_t;

typedef struct {
    point_t loc;
    double tran_power;
    double frequency;
} wap_t;

typedef struct {
    point_t p1;
    point_t p2;
} line_t;

typedef struct {
    int n;
    wap_t a[MAX_ELE_NUM];
} wap_list_t;

typedef struct {
    int n;
    point_t a[MAX_ELE_NUM];
} observe_list_t;

typedef struct {
    int n;
    point_t a[MAX_ELE_NUM];
} boundary_list_t;


/*======================================================================*/
/* Functions used in this program */
void stage(int i);
double strength(wap_t *p,  double x, double y);
double maximum_strength(wap_list_t *pwl,  double x, double y);
void map_symbol(double strength);
int line_intersect(line_t l1, line_t l2);


/*======================================================================*/
/* Main body of the program */
int main(int argc, char *argv[]){

    /*====================================================================*/
    /* Declare variables and initializations */
    int i,j,k,se_flag;
    double strength,center_xsum,center_ysum;
    char flag;
    wap_list_t wl;
    observe_list_t ol;
    boundary_list_t bl;
    line_t l1,l2;

    center_xsum=0;
    center_ysum=0;
    wl.n = 0;
    ol.n = 0;
    bl.n = 0;


    /*====================================================================*/
    /* Read data and store them */
    while(scanf("%c", &flag) == 1){
        if(flag == 'W'){
            /* The other way to read data:
            scanf("%lf %lf %lf %lf", &wl.a[wl.n].loc.x,
            &wl.a[wl.n].loc.y,&wl.a[wl.n].tran_power,
            &wl.a[wl.n].frequency); */
            wap_t *p_on = &wl.a[wl.n];
            scanf("%lf %lf %lf %lf", &p_on->loc.x,&p_on->loc.y,
                &p_on->tran_power,&p_on->frequency);
            wl.n++;
        }
        if(flag == 'P'){
            /* The other way to read data:
            scanf("%lf %lf", &ol.a[ol.n].x,&ol.a[ol.n].y); */
            point_t *p_tw = &ol.a[ol.n];
            scanf("%lf %lf", &p_tw->x,&p_tw->y);
            ol.n++;
        }
        if(flag == 'V'){
            /* The other way to read data:
            scanf("%lf %lf", &bl.a[bl.n].x,&bl.a[bl.n].y); */
            point_t *p_th = &bl.a[bl.n];
            scanf("%lf %lf", &p_th->x,&p_th->y);
            bl.n++;
        }
    }


    /*====================================================================*/
    /* Count WAPs' number and print out the max signal strength at origin*/
    stage(1);
    printf("Number of WAPs: %02d\n", wl.n);
    printf("Maximum signal strength at (%04.1f, %04.1f): %05.2f dBm\n",
        ORIGIN_COOR_X, ORIGIN_COOR_Y,
        maximum_strength(&wl, ORIGIN_COOR_X, ORIGIN_COOR_Y));
    printf("\n");


    /*====================================================================*/
    /* Print out the max signal strength at observed points*/
    stage(2);
    for(k=0;k<ol.n;k++){
    printf("Maximum signal strength at (%04.1f, %04.1f): %05.2f dBm\n",
        ol.a[k].x, ol.a[k].y,
        maximum_strength(&wl, ol.a[k].x, ol.a[k].y));
    }
    printf("\n");


    /*====================================================================*/
    /* Sample points and compare the max signal strengths of
      them with a specific signal strength value */
    stage(3);
    k = 0;
    for(i=0;i<REGION_LEN-1;i++){
        for(j=0;j<REGION_LEN-1;j++){
            strength = maximum_strength(&wl, j+1, REGION_LEN-1-i);
            if(strength <= S3_TESTED_STR){
                k++;
            }
        }
    }
    printf("%04d points sampled\n", (int)pow(REGION_LEN-1,2));
    printf("%04d points (%05.2f%%) with maximum signal strength <= %d dBm\n",
        k,k*100.0/pow(REGION_LEN-1,2),S3_TESTED_STR);
    printf("\n");


    /*====================================================================*/
    /* Calculate the max signal strength of each cell and draw signal map*/
    stage(4);
    for(i=0;i<REGION_LEN/2;i++){
        for(j=0;j<REGION_LEN;j++){
            strength = maximum_strength(&wl,0.5+j,REGION_LEN-1-2*i);
            map_symbol(strength);
        }
        printf("\n");
    }
    printf("\n");


    /*====================================================================*/
    /* Consider about boundaries and draw signal strength map */
    stage(5);
    for(k=0;k<bl.n;k++){
        center_xsum += bl.a[k].x;
        center_ysum += bl.a[k].y;
    }
    l1.p1.x = center_xsum/bl.n;
    l1.p1.y = center_ysum/bl.n;
    for(i=0;i<REGION_LEN/2;i++){
        for(j=0;j<REGION_LEN;j++){
            se_flag=0;
            l1.p2.x=0.5+j;
            l1.p2.y=REGION_LEN-1-2*i;
            for(k=0;k<bl.n;k++){
                l2.p1=bl.a[k];
                l2.p2=bl.a[(k+1)%bl.n];
                if(line_intersect(l1, l2)){
                    se_flag = 1;
                }
            }
            if (se_flag){
                printf("#");
            } else{
                strength = maximum_strength(&wl, l1.p2.x, l1.p2.y);
                map_symbol(strength);
            }
        }
        printf("\n");
    }
    return 0;
}


/*======================================================================*/
/* Print the title of every stage */
void stage(int i){
    printf("Stage %d\n", i);
    printf("==========\n");
}


/*======================================================================*/
/* Calculate the signal strength provided by the WAPs at one point  */
double strength(wap_t *p,  double x, double y){
    double distance,fspl,strength;
    distance = sqrt(pow(p->loc.x - x, 2)+pow(p->loc.y - y, 2));
    fspl = 20*log10(distance)+20*log10(p->frequency)+32.45;
    return strength = p->tran_power - fspl;
}


/*======================================================================*/
/* Return the maximum signal strength at the point */
double maximum_strength(wap_list_t *pwl,  double x, double y){
    int i;
    double signal_str, max_strength;
    max_strength= strength(&(pwl->a[0]), x, y);
    for(i=1;i < pwl->n;i++){
        signal_str = strength(&(pwl->a[i]),x,y);
        if (signal_str > max_strength){
            max_strength = signal_str;
        }
    }
    return max_strength;
}


/*======================================================================*/
/* Use symbols to represent the max signal strength at one point */
void map_symbol(double strength){
    if(strength >0){
        printf("%c",STR_POS);
    }
    if(-10<strength && strength<=0){
        printf("%c",STR_NEG_TEN);
    }
    if(-20<strength && strength<=-10){
        printf("%c",STR_NEG_TWE);
    }
    if(-30<strength && strength<=-20){
        printf("%c",STR_NEG_THI);
    }
    if(-40<strength && strength<=-30){
        printf("%c",STR_NEG_FOR);
    }
    if(-50<strength && strength<=-40){
        printf("%c",STR_NEG_FIF);
    }
    if(strength<= -50){
        printf("%c",STR_NEG_SMA_FIF);
    }
}


/*======================================================================*/
/* Test whether the projections of the two line segments intersect or not */
/* This function is creatived by the other author */
int line_intersect(line_t l1, line_t l2) {
  double x1=l1.p1.x, y1=l1.p1.y,
      x2=l1.p2.x, y2=l1.p2.y,
      x3=l2.p1.x, y3=l2.p1.y,
      x4=l2.p2.x, y4=l2.p2.y;
  double mua,mub;
  double denom,numera,numerb;

  /* Take the projections of the two line segments */
  double xMin1, xMax1, xMin2, xMax2, yMin1, yMax1, yMin2, yMax2;
  xMin1 = MIN(x1, x2);
  xMax1 = MAX(x1, x2);
  xMin2 = MIN(x3, x4);
  xMax2 = MAX(x3, x4);

  yMin1 = MIN(y1, y2);
  yMax1 = MAX(y1, y2);
  yMin2 = MIN(y3, y4);
  yMax2 = MAX(y3, y4);

  /* Do the projects intersect? */
  if((xMin2-xMax1) >= EPS || (xMin1-xMax2) >= EPS ||
      (yMin2-yMax1) >= EPS || (yMin1-yMax2) >= EPS) {
      return FALSE;
  }

  denom  = (y4-y3) * (x2-x1) - (x4-x3) * (y2-y1);
  numera = (x4-x3) * (y1-y3) - (y4-y3) * (x1-x3);
  numerb = (x2-x1) * (y1-y3) - (y2-y1) * (x1-x3);

  /* Are the line coincident? */
  if (ABS(numera) < EPS && ABS(numerb) < EPS && ABS(denom) < EPS) {
    return(TRUE);
  }

  /* Are the line parallel */
  if (ABS(denom) < EPS) {
    return(FALSE);
  }

  /* Is the intersection along the the segments */
  mua = numera / denom;
  mub = numerb / denom;
  /* AM - use equality here so that "on the end" is not an
   * intersection; use strict inequality if "touching at end" is an
   * intersection */
  if (mua < 0 || mua > 1 || mub < 0 || mub > 1) {
    return(FALSE);
  }
  return(TRUE);
}

/*======================================================================*/
/* C programming is fun */
```