```c

#include <math.h>
#include "tiff.h"
#include "allocate.h"
#include "randlib.h"
#include "typeutil.h"

void error(char *name);

// sort array
void sort(int *pixel, int *weight, int length)
{
    int i, j;
    int v;
    for(i = 0; i < length - 1; i ++)
        for(j = i+1; j < length; j ++)
        {
            if(pixel[i] < pixel[j])
            {
                v = pixel[i];
                pixel[i] = pixel[j];
                pixel[j] = v;

                v = weight[i];
                weight[i] = weight[j];
                weight[j] = v;
            }
        }
}

// input a pixel in a image, output the weighted median
int WMfilter(struct TIFF_img input_img, int x, int y)
{
    int pixel[25]; // pixels by 5*5 filter
    int weight[25] = {1,1,1,1,1,1,2,2,2,1,1,2,2,2,1,1,2,2,2,1,1,1,1,1,1}; //
weighted factors
    int i, j, k=0;
    // int istar; // output
    // get 5*5 window from image
    for (i=x-2; i<=x+2; i++)
    {
        for (j=y-2; j<=y+2; j++)
        {
            pixel[k] = input_img.mono[i][j];
            k = k + 1;
        }
    }
    // sort the pixel and weight
    sort(pixel,weight,25);
    // find i*
    int temp = 0;
    for (i=0; i<25; i++)
    {
        temp = temp + weight[i];
        if (temp >= 17)
        {
            break;
        }
    }
    return pixel[i];
```

```
}



int main (int argc, char **argv)
{
    FILE *fp;
    struct TIFF_img input_img, output_img;
    // double **img1,**img2;
    int32_t i,j;

    printf("processstart");

    if ( argc != 2 ) error( argv[0] );

    /* open image file */
    if ( ( fp = fopen ( argv[1], "rb" ) ) == NULL ) {
    fprintf ( stderr, "cannot open file %s\n", argv[1] );
    exit ( 1 );
    }

    /* read image */
    if ( read_TIFF ( fp, &input_img ) ) {
    fprintf ( stderr, "error reading file %s\n", argv[1] );
    exit ( 1 );
    }

    /* close image file */
    fclose ( fp );

    /* check the type of image data */
    if ( input_img.TIFF_type != 'g' ) {
    fprintf ( stderr, "error:  image must be grey scale\n" );
    exit ( 1 );
    }

    /* Allocate image of double precision floats */
    // img1 = (double **)get_img(input_img.width,input_img.height,sizeof(double));
    // img2 = (double **)get_img(input_img.width,input_img.height,sizeof(double));

    /* set up structure for output achromatic image */
    /* to allocate a full color image use type 'c' */
    get_TIFF ( &output_img, input_img.height, input_img.width, 'g' );

    /*
    // copy image component to double array
    for ( i = 0; i < input_img.height; i++ )
    for ( j = 0; j < input_img.width; j++ ) {
    img1[i][j] = input_img.mono[i][j];
    }
    */

    /* Filter image */
    for ( i = 2; i < input_img.height-2; i++ )
    for ( j = 2; j < input_img.width-2; j++ ) {
    output_img.mono[i][j] = WMfilter(input_img,i,j);
    }

    /*fill in boundary pixels*/
    for ( i = 0; i < 2; i++ )
```

```
    for ( j = 2; j < 2; j++ ) {
    output_img.mono[i][j] = 0;
    }
    for ( i = input_img.height-2; i < input_img.height; i++ )
    for ( j = input_img.width-2; j < input_img.width; j++ ) {
    output_img.mono[i][j] = 0;
    }
    for ( i = 0; i < 2; i++ )
    for ( j = input_img.width-2; j < input_img.width; j++ ) {
    output_img.mono[i][j] = 0;
    }
    for ( i = input_img.height-2; i < input_img.height; i++ )
    for ( j = 2; j < 2; j++ ) {
    output_img.mono[i][j] = 0;
    }

    /* open output image file */
    if ( ( fp = fopen ( "output.tif", "wb" ) ) == NULL ) {
    fprintf ( stderr, "cannot open file output.tif\n");
    exit ( 1 );
    }

    /* write output image */
    if ( write_TIFF ( fp, &output_img ) ) {
    fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
    exit ( 1 );
    }

    /* close output image file */
    fclose ( fp );

    /* de-allocate space which was used for the images */
    free_TIFF ( &(input_img) );
    free_TIFF ( &(output_img) );
    //free_TIFF ( &(color_img) );

    //free_img( (void**)img1 );
    //free_img( (void**)img2 );

    printf("processsuccess");

    return(0);
}

void error(char *name)
{
    printf("usage:  %s  image.tiff \n\n",name);
    printf("this program reads in a 24-bit color TIFF image.\n");
    printf("It then horizontally filters the green component, adds noise,\n");
    printf("and writes out the result as an 8-bit image\n");
    printf("with the name 'green.tiff'.\n");
    printf("It also generates an 8-bit color image,\n");
    printf("that swaps red and green components from the input image");
    exit(1);
}
```