HW06 - Due Tu 8 March 2016 before 11:59 PM

Dr. Touma

March 2, 2016

Use HW05 as basis and break up your code into several source files and a make file. The output of the code should be the same as HW05. If your HW05 has errors, correct them before proceeding or you can use my HW05 code as basis. You must follow these guidelines for this homework assignment.

1. your source directory must contain 4 files with the following names: *main.c*, *class_stats.c*, *class_stats.h*, and *Makefile*

2. *main.c* should contain the *main* function and the appropriate headers. Inside *main* you'll issue calls to all other functions after you read in the number of assignments and number of students and do the proper array/structure initialization. You need to also check for the total count after binning before proceeding. In other words there is no need to compute statistics if binning is not successful.

3. *class_stats.h* should contain only the structure for the student statistics (as in HW05) and prototypes of the functions as shown in the listing below.

4. *class_stats.c* should contain the implementation of the functions prototyped in *class_stats.h*.

5. instead of the variance, display the standard deviation which is the square root of the variance.

6. we'll discuss *Makefile* in class on Thursday

```
#ifndef CLASS_STATS
#define CLASS_STATS

// structure that holds the statistics for an assignment
typedef struct
{
   int min , max , num_of_students ;
   float mean , median , std_dev ;
} Statistics ;

// sorts the values of an array in ascending order
// input are the array to be sorted and the number of elements in the array
void sort_a ( int *array , int num_elements ) ;

// calculates the mean of the elements of an array
// input are the array of data and the number of elements in the array; returns the mean
float calculate_mean ( const int *array, int num_elements );

// calculates the variance of the emelemts of an array
// input are the array of data and the number of elements in the array
// this function calls calculate_mean; returns the variance
float calculate_variance ( const int *array , int num_elements );

// calculates the median of the elemets of an array
// input are the array of data and the number of elements in the array
// the input array is not necessarily sorted; it calls sort_a first to sort a copy of the array then
// computes the median and returns it
float calculate_median ( const int *array , int num_elements );

// finds the maximum value of the elements of an array
// input are the array of data and the number of elements in the array; returns the maximum element
int calculate_max ( const int *array , int num_elements );

// finds the minimum value of the elements of an array
// input are the array of data and the number of elements in the array; returns the minimum element
int calculate_min ( const int *array , int num_elements );

// gets the grades of n students for an assignment from the user
// input are the array that hold the grades and the number of students; it modifies the array
void get_user_input ( int *array , int num_elements );

// bin the grades
// input are the grades array to be binned and its size; the arrays holding the binned grades and its
    size
// returns the total number of grades binned
```

```
int bin_grades ( int num_students , const int *grades ,int assignment , int grades_scale [11][
      assignment]) ;

// get stats
// input are the grades array and its size along with the statistics structure and the assignment that
      the stats
// are being computed for. the structure gets modified inside the function
void get_stats ( const int *grades , int num_elements, int assignment, Statistics *stats );

// print the grades
// input are the binned array and its size and the statistics structure and its size
void display_grades_distribution ( int num_assignments , const int grades_scale [11][num_assignments] ,
       Statistics *stats );

#endif
```