Contest Task Description and Accompanying Code

Description:

You have to complete **ten** tasks for this contest. Some tasks have more than one function to complete. The tasks are summarized below:

Grading and Points Summary:

Following table summarizes the grading system. You will have to complete all the functions successfully to obtain a perfect score of 125/125.

Task No.	Task Name	Function Name	Total Marks	Break Down
1	Temperature Convert	`far2cel(), cel2far()`	10	5 + 5
2	Find Mean	`get_mean()`	5	5
3	Find Odd Todd	`odd_todd()`	10	10
4	Find Variance	`get_variance()`	10	10
5	Find LCM	`get_lcm()`	10	10
6	Test Prime Factor	`isPrimeFactor()`	10	10
7	Sort Numbers	`sort_list(), find_min(), swap()`	20	10 + 5 + 5
8	Find Median	`get_median`	10	10
9	Get Unique List	`get_unique_list()`	20	20
10	Find Size of File	`get_file_size()`	20	20
		Total →	125	

Accompanying Code

You are provided with a zip file (ContestCode.zip) containing the necessary files for the sample contest. The zip file contains a CodeBlocks project with the following C files:

- main.c
- tasks.c (You will only insert code in this file)
- tasks.h
- auto_grader.o
- auto_grader.h

If you use some other IDE (e.g. DevC) you may make a new project and add the above listed files to the project.

The file '***tasks.c***' provides you with skeleton code for the Programming Competition. It contains ten tasks that you will have to complete. You will insert your code only in the designated areas leaving the rest of the code unchanged.

If you decide to make any ***new functions*** make sure that:

1. Their prototypes are added in the tasks.h file.
2. These new functions MUST be called from within the existing functions. (Refer to code for Task 6 for an example).

Detailed explanation of the tasks is given below.

Task1: Convert temperature from Celsius to Fahrenheit and vice-versa. You will complete two functions for this task.

a) Write a C function `'float cel2far(float cel);'` that takes temperature in Celsius as input and returns the converted value in Fahrenheit.
b) Write a C function `'float far2cel(float far);'` that takes temperature in Fahrenheit as input and returns the converted value in Celsius.

Task 2: Find 'Arithmetic Mean' of a given array of integers.

Write a C function with the following prototypes

```
float get_mean(int * ptr, int size);
```

The function takes two inputs. The first input (`int * ptr`) is a pointer to the input array. The second argument is the array size (`int size`). Your code should find the Arithmetic Mean (Sum divided by Number of Elements) of all the numbers in this array and return it.

Note: The return type of this function is a float.

Task 3: Find the number of odd-todds in a given array of integers.

An odd-todd is a sequence of three consecutive odd numbers in an array. Your task is to find and return the number of such instances present in the input array. You will write a function with the following prototype

```
int odd_todd(int * num_array, int size);
```

The function takes two inputs. The first input (`int * ptr`) is a pointer to the input array. The second argument is the array size (`int size`).

Example: Input Array [2, 3, 5, 7, 4, 21, 23, 5, 9, 6]

This array has following 3 odd-todds:

- 3, 5, 7
- 21, 23, 5
- 23, 5, 9

Therefore the function when given this array as input should return 3.

Task 4: Calculate the variance of data in a given array of integers.

In this task you will compute the Variance of a given set of integer values. Implement the formula given in the following figure to compute the Variance.

You will insert your code in the following function:

```
float get_variance(int * ptr, int size)
{
      /* Insert your code here */
}
```

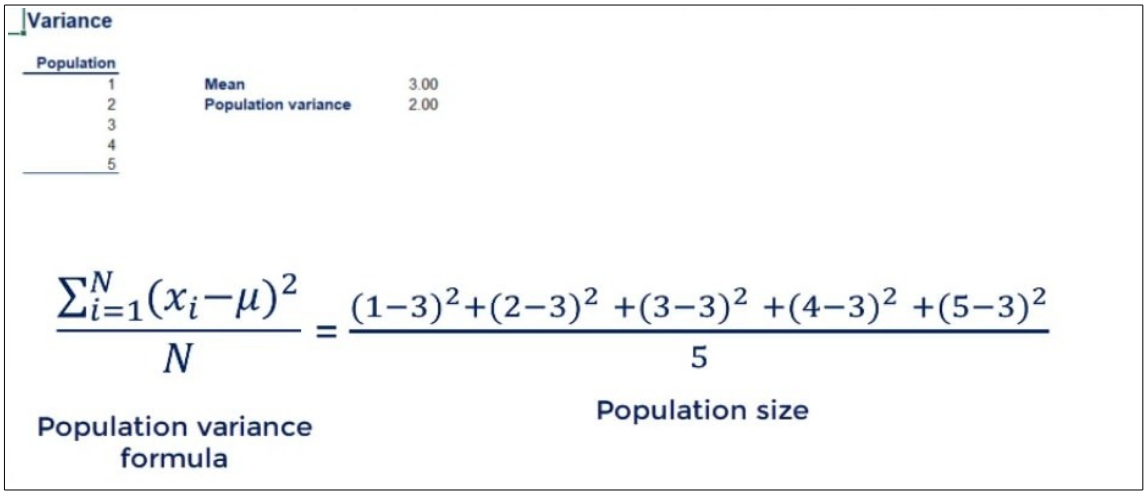

Figure 1 Calculating Population Variance

The function takes two inputs. The first input (`int * ptr`) is a pointer to the input array. The second argument is the array size (`int size`). The function should compute the variance and return the floating point answer.

Hint: You can call the `get_mean()` function completed in Task 2.

Task 5: Calculate LCM of a given array of numbers.

Your task is to compute the Least Common Multiple (LCM) of the numbers (+ve integers) given in the input array. You will complete the code in the function with the following prototype:

```
int get_lcm(int * ptr, int size);
```

The function takes two inputs. The first input (`int * ptr`) is a pointer to the input array. The second argument is the array size (`int size`). The function should compute the LCM and return the integer answer.

Task 6: Test if a given divisor is a Prime Factor of a number.

For this task you will insert your code in the function:

```
int isPrimeFactor(int input, int divisor);
```

The function tells whether the given divisor (`int divisor`) is a Prime Factor of the input number (`int input`). The function returns (1) if the divisor is a Prime Factor, (0) otherwise.

Task 7: Sort an integer array in Ascending order

Your task is to complete the functions with the following prototypes;

a) `int find_min(int * ptr, int size);`
   // Function computes and returns the ***index*** of the smallest element in the array.
b) `void swap(int * num1, int * num2);`
   // Function swaps the values of variables pointed to by their respective pointers.
c) `void sort_list(int * ptr, int size);`

This function takes as input a pointer to the start of the array, and the array size and sorts it in-place.

Note: You can call the functions (**find_min**() and **swap**()) to implement Selection Sort algorithm.

Task 8: Find the Median from an array of numbers.

The **median** is the number that is halfway into the set. To find the **median**, the data should be arranged in order from least to greatest. If there is an even number of items in the data set, then the **median** is found by taking the mean (average) of the two middlemost numbers.

You will complete the function with the following prototype.

```
int get_median(int * ptr, int size);
```

The function takes two inputs. The first input (`int * ptr`) is a pointer to the input array. The second argument is the array size (`int size`). The function should find the Median and return the ***integer*** answer.

Note: The Median of an array with even number of elements will be the ***integer*** average of the two middle numbers in this implementation.

Task 9: Get unique number list from a given array of integers

```
void get_unique_list(int * ptr_src, int * ptr_dst, int size, int * size_unq);
```

The task is to find the number of of unique elements in an array (**ptr_src**) and copy them to another array (**ptr_dst**). Both these arrays will be initialized in the calling function. The function should also return (by reference) the number of unique elements found in (**size_unq**)

Example:

for input_array [2, 2, 3, 4, 2, 5, 6, 3, 7, 6, 5, 3, 2]

the destination_array will be [2, 3, 4, 5, 6, 7] and (* size_unq = 6).

Task 10: Calculate the size of a given file in kBs.

In this task you will complete the function with the following prototype:

```
float get_file_size(char * filename);
```

The function takes the file name (address to the start of a NULL terminated character array) as input. The function should then open the file and find the number of bytes it contains till EOF. The number of bytes divided by 1024 will give the size in kBs. If the file cannot be opened the function should return -1.

****** End of Document ******