Offline - 2: Arrays

Full Marks: 100 **Deadline: 26 May, 2025 11:59 P.M**

Problem Overview

In this task, you will write a **C program** that allows a user to interactively analyze an array of real numbers using basic statistics. The data in the array will be generated using either a **uniform** or **Gaussian (normal)** distribution. The program will support commands to initialize the array, compute and display basic statistics (min, max, mean, standard deviation), and show a histogram. The program runs in an **interactive loop**, taking one command at a time and responding appropriately.

Program Requirements

1. Global Constant and Array Declaration

- Define constants:

```
#define MAX_SIZE 1000000
#define MAX_BINS 1000
#define MAX_STARS 50
```

- Declare an array large enough to hold `MAX_SIZE` elements:

```
double array[MAX_SIZE];
```

2. Program Flow

The program should begin by printing a list of supported commands (like a help message). Then, it should enter a loop that repeatedly prompts the user for a command.

3. User Commands

The following table lists valid commands and their behavior:

Command	Description
`set`	Ask user for number of elements, seed, and distribution type (`uniform` or `gaussian`). Then ask for parameters and populate the array using provided functions.
`min`	Print the **minimum** value in the array.
`max`	Print the **maximum** value.
`mean`	Print the **average** of the elements.
`stddev`	Print the **standard deviation**.
`summary`	Print all 4 statistics: min, max, mean, stddev.
`hist`	Ask user for number of bins, and then display a histogram.
`help`	Print the list of all available commands.
`exit`	Exit the program.

4. Behavior Details

A. `set`

- Ask for:
  - Number of elements (must be $\leq$ `MAX_SIZE`).
  - A seed (unsigned int). Basically this ensures that the random numbers generated are the same every time you run the program with the same seed. You don't need to worry too much about this.
  - Distribution type: `uniform` or `gaussian`.
- If `uniform`, also ask for:
  - Minimum and maximum values.
- If `gaussian`, ask for:
  - Mean and standard deviation.
- Use the following functions to populate the array. They have been provided to you in the `utils.c` file:

```
populate_array_uniform(array, size, min, max, seed);
populate_array_gaussian(array, size, mean, stddev, seed);
```

  Ensure that your `main.c` file and the `utils.c` file is in the same folder. You may use the provided functions by including the header file:

```
#include "utils.c" // Use "double quotes". Not <angle brackets>.
```

  Please examine the file contents and understand their usage. In case of any problems, just copy the contents of `utils.c` file to your `main.c` file.
- If user enters an invalid distribution, print:

```
Invalid distribution.
```

B. `min`, `max`, `mean`, `stddev`

- Traverse the array manually using loops.
- Do **not** use library functions (except `sqrt` from `<math.h>`).
- If the array is not yet initialized with `set`, print:

```
Array not initialized. Use 'set' command first.
```

- Print results with exactly 4 digits after the decimal, like this:

```
Min       :    -3.2462
Max       :     9.8321
Mean      :     1.5837
Std Dev   :     2.1082
```

- Labels (`Min`, `Max`, `Mean`, `Std Dev`) should be left-aligned, and values should be right-aligned using `printf` format specifiers.
- Use proper spacing and alignment in the output so that it looks neat and organized. Try to match your output with the provided sample output as closely as possible. Use the same seed and parameters as in the sample output, and you can compare your output with the provided sample output in this website: `https://www.diffchecker.com/`.

C. `hist`

- Ask for number of bins (e.g., 10). The number of bins must be ≤ 1000. You may use a pre-defined `MAX_BINS` constant.
- If the user enters an invalid bin count, print:

```
Invalid number of bins. Must be between 1 and 1000.
```

- Compute:

```
bin_width = (max - min) / bins;
```

- Create an `int` array of size `bins` to hold counts. You may use a pre-declared:

```
int bin_counts[MAX_BINS];
```

- Traverse each element and compute:

```
bin_index = (int)((value - min) / bin_width);
```

- Count how many values fall into each bin.
- For each bin, print the counts as `*` symbols, normalized to a maximum of `MAX_STARS` `*` symbols per bin (scale based on highest bin count). Example:

```
[ 0.00 -  1.00]: ********************
[ 1.00 -  2.00]: *********
[ 2.00 -  3.00]: ********
.....
```

D. `help`

Print this list of commands:

```
Commands:
  set     - Set array size, seed, and distribution (uniform or gaussian)
  min     - Print minimum value
  max     - Print maximum value
  mean    - Print mean value
  stddev  - Print standard deviation
  hist    - Plot histogram
  summary - Print min, max, mean, stddev
  help    - Show this help message
  exit    - Exit the program
```

E. `exit`

End the loop and terminate the program.

5. Unknown Commands

If the user enters a command not in the list, print:

```
Unknown command. Type 'help' for list of commands.
```

Make sure that before performing any operation, you check if the array has been initialized with the `set` command. If not, print:

```
Array not initialized. Use 'set' command first.
```

Bonus Task (10 Marks)

Implement a new command, `median`, that computes and displays the median value of the array. To calculate the median, sort the array in ascending order using a the `qsort` function. If the array has an odd number of elements, the median is the middle element; if even, it is the average of the two middle elements.

Sample Input/Output

Refer to the provided `.txt` files for sample input and output.

Mark Distribution

Component	**Marks**
`set` command with correct prompts	20
Computation of statistics (min, max, mean, stddev)	20
Histogram logic and bin counting	20
Proper formatting of output	15
Help message	5
Handling of unknown commands	10
Proper indentation and readable code	10
Bonus task (median)	10
Total	**110**

Submission Guidelines

Put your `main.c` and `utils.c` files in a folder named `2405ABC`. Zip the folder and submit it on Moodle. Make sure to test your program thoroughly before submission.

Blindly copying *from other* ***students or any other source*** *(chatGPT, internet etc.) will result in a* ***-100%*** *for the assignment.* The assignment is meant to be a learning experience. You may discuss with your classmates, but please do not copy their code.