**CS 159 – HW #07**
**Due: Monday April 26, 2021 at 11:00pm** (time local to West Lafayette, IN).
**10 Points Possible**

**Background:** Given a positive integer the following can be applied repeatedly and eventually the number 1 will be reached; If the integer value is odd then multiply it by three and add 1, if the integer value is even then divide it by 2.

- Example starting with the integer twenty: $20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

**Problem:** The user will specify whether they wish the data set to be (1) sorted by the count of the number of applications of the above steps necessary to reach 1 or (2) sorted by the first power of two encountered during the application of the above steps. The positive integer data set will follow that includes at least one value with a maximum number of 35 values.

- Example starting with the integer twenty: (1) count of seven applications of change to reach 1 and (2) the first power of two encountered is 16.

**Example Execution #1:**

```
Enter sorting option (1) count or (2) power of two -> 2
Enter up to 35 integer values -> 600 300 700 100 -1

Sorted data by power of two: 100 700 300 600
```

**Example Execution #2 (same data set as first execution, different order of input, same output generated):**

```
Enter sorting option (1) count or (2) power of two -> 2
Enter up to 35 integer values -> 700 100 600 300 -1

Sorted data by power of two: 100 700 300 600
```

**Example Execution #3:**

```
Enter sorting option (1) count or (2) power of two -> 1
Enter up to 35 integer values -> 10 20 30 40 50 60 70 -1

Sorted data by count: 10 20 40 70 30 60 50
```

**Example Execution #4:**

```
Enter sorting option (1) count or (2) power of two -> 1
Enter up to 35 integer values -> 27 12 19 15 9 14 7 18 -1

Sorted data by count: 12 7 14 15 9 18 19 27
```

**Example Execution #5:**

```
Enter sorting option (1) count or (2) power of two -> 0

Error! Enter option 1 or 2 only!

Enter sorting option (1) count or (2) power of two -> 3

Error! Enter option 1 or 2 only!

Enter sorting option (1) count or (2) power of two -> 1
Enter up to 35 integer values -> 41 35 89 71 113 145 55 31 73 43 -1

Sorted data by count: 113 35 43 89 71 31 41 55 73 145
```

---

**All course programming and documentation standards are in effect for this and each assignment this semester. Please review this document!**

---

**Example Execution #6 (the -1 terminal value is not needed when the maximum input is given):**

```
Enter sorting option (1) count or (2) power of two -> 2
Enter up to 35 integer values -> 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19
18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Sorted data by power of two: 1 2 4 8 3 5 6 7 9 10 11 12 13 14 15 16 17 18 19 20 22 23
24 25 26 27 28 29 30 31 33 34 35 32 21
```

**Academic Integrity Reminder:**

- Please review the policies of the course as they relate to academic integrity. The assignment you submit should be your own original work. You are to be consulting only course staff regarding your specific algorithm for assistance. Collaboration is not permitted on individual homework assignments.

**Additional Requirements:**

1. Add the homework assignment header file to the top of your program. A description of your program will need to be included in the assignment header.
2. **Each of the example executions provided for your reference represents a single execution of the program.** Your program must accept input and produce output **exactly** as demonstrated in the example executions, do not add any "bonus" features not demonstrated in the example executions. Your program may be tested with the data seen in the example executions and an unknown number of additional tests making use of meaningful data as specified in the problem description.
   - Example execution #5 demonstrates the only input validation requirements for this assignment.
3. For this assignment you will be **required** to implement the user-defined functions (from chapter 4). Failing to follow course standards as they relate to good user-defined function use will result in a **zero for this assignment.** Good function design will limit each function to a single task, eliminate redundant logic in the program, and maximize re-use of functions within a program.
4. Revisit **course standards as it relates what makes for good use of user-defined functions, what is acceptable to retain in the** `main` **function, and when passing parameters by address is appropriate.** In many cases user-defined function use should result in a `main` function that only declares variables and makes calls functions. Selection and repetition constructs in a `main` function are to facilitate the calling of functions.
5. Course standards **prohibit** the use of programming concepts not yet introduced in lecture. For this assignment you can consider all material in the **first ten chapters** of the book, notes, and lectures to be acceptable for use.
6. A program **MUST** compile, be submitted through the `guru` server prior to the posted due date to be considered for partial credit. The submission script will reject the submission of any file that does not successfully compile on the `guru` server. The name of the source code file you attempt to submit must be `hw07.c`, no variation is permitted.

**Selected Course Programming and Documentation Standards Reminders:**

- It is common to make use of a symbolic/defined constant when the size of the array is known prior to the start of a program.
- Code found inside the body of relevant selection and repetition constructs must be indented two additional spaces.
- Make use of `{` and `}` with all relevant selection and repetition constructs.
- See page 258 of your C programming text regarding the proper indentation for a `switch` construct.
- Use the course function header (`head_fx vi` shortcut `hfx` while in command mode) for every user-defined function in your program.
  - List and comment **all parameters** to a function, one per line, in the course function header.
  - **All function declarations** will appear in the global declaration section of your program.
  - **The user-defined function definitions will appear in your program after the** `main` **function.**
- Maximize your use of symbolic/defined constants and minimize your use of literal constants.
- Comment **all** variables to the right of each declaration. Declare only one variable per line.

**Selected Course Programming and Documentation Standards Reminders (continued):**

- In general it is acceptable to initialize a variable declared in the local declaration section of a function. If the expression used to initialize the variable is more complex than a constant assignment then it is best to give the variable its first value inside of the executable statement section of the function.
- At no point during the semester should the local declaration and executable statement sections of a function ever overlap.
- Select **meaningful identifiers** (names) for all variables in your program.

**When you submit...** only the final successful submission is kept for grading. All other submissions are over-written and cannot be recovered. You may make multiple submissions but only the last attempt is retained and graded.

- Verify in the confirmation e-mail sent to you by the course that you have submitted the correct file (must be named `hw07.c`), to the correct assignment (`hw07`), and to the correct section.
- **Leave time prior to the due date to seek assistance** should you experience difficulties completing or submitting this assignment. All attempts to submit via a method other than through the `guru` server as set up in the Account Configuration Activity will be denied consideration.

**Assignment deadlines...** are firm and the electronic submission will disable promptly as advertised. We can only grade what you submit as expected prior to the assignment deadline.

**Auto-Grade Tool**

- We have implemented what is being referred to as the auto-grade tool. At the time of a successful assignment submission you may receive some feedback on your program in regards to course programming and documentation standards. This feedback may include a potential deduction that you will receive once your assignment is reviewed by your grader.
- It is expected that graders verify those notes identified by this tool to ensure that they are indeed applicable and reasonable to the submission. Graders may make additional deductions for those standards not identified by the new tool.
- We hope that this feedback helps with the enforcement of course standards, consistency in grading across sections, and to encourage students to revise their work when problems are identified before the assignment deadline passes. It is possible to resubmit an assignment for grading up to the advertised deadline. Only the final successful submission is retained and evaluated.