College of

Nanotechnology, Sciences, and Engineering Department of Computer Science

ICSI 333 SYSTEM FUNDAMENTALS CQUPT, SPRING 2025

LAB 2.1 - 2.3

The total grade for three labs is 100 points.

You must follow the programming and documentation guidelines.

**DESCRIPTION**

You must write a C program that generates random numbers in the range of [-100.0, 100.0] and, depending on the value of the number, creates a new node at the beginning of one of the four linked lists: positive numbers with a non-zero whole part, positive fractions, negative fractions, and negative numbers with the non-zero whole part. The program stops when the total amount of randomly generated positive numbers reaches the value specified by the user.

The nodes of the linked lists must keep the generated number, the structure with a sign, an exponent, and a mantissa of IEEE 754 representation of the number, the value restored from the IEEE 754 representation, and a pointer to the next node.

If the restored value is not equal to the value of the generated number, the node should not be added, and an error message should be printed to stderr.

Finally, each linked list must be saved in a separate file, but all four files must have the same extension specified by the user. Special requirements:

* The user preferences must be inputted as command-line arguments.
* You should use the results of Project 1 as a library function that returns the structure with the sign, exponent, and mantissa.
* To calculate the restored value, you should write and add to the library a function that implements the following formula VALUE = -1*S* × 1.*M* × 2*E*-127, where *S* is the sign bit (0 or 1), *M* is the mantissa (000...0002 to 111...1112) and *E* is the biased exponent (000000002 to 1111 11102).
* To get full credit for the project, it must be organized into three files: the main program, the library module with implementations of IEEE 754 conversion functions, and the header file.

Work on the project task by task.

**RECOMMENDED STEPS OF DEVELOPMENT**

TASK #1. LINKED LISTS

Start with the generating of random numbers with the function rand(). You may find an example in the lecture handouts. But this time the result should be a float number in the range of [-100.0, 100,0].

At this stage, you can hardcode the desired quantity of positive numbers. I recommend setting it to at least 100, so your program can generate numbers in each of the four categories given in the description.

To add a new node to the proper linked list you may use the lecture handout or textbook examples but note that the new nodes must be added at the beginning of the lists.

TASK #2. FILES

At this stage, you can hardcode the file names and simplify the nodes, so they hold only a number and a pointer. Your program must save four linked lists in four different files.

TASK #3. COMMAND LINE ARGUMENTS

Function main() can take string arguments from the command line. Your program must be run with two arguments: the desired quantity of positive numbers to stop random generating and the files’ extension.

You must consider the situation when the user misses the arguments. It will be your choice to use the default values or stop execution with a message to the user.

The example of command line arguments you can find in the textbook or the handouts to the lecture about files.

TASK #4. LINKING TO PROJECT 1 FROM LAB 1

Modify your Project 1, so it has no main() function but a collection of functions including one that receives a real number and returns a structure that holds IEEE 754 representation as sequences of 0s and 1s with the fields: sign, exponent, mantissa. Use typedef to define this structure as a new type in a header file. The header file must have the same name as the file with functions from Project 1 and the extension .h. The header must also have the prototypes of all functions that can be used by the third file. For example, if your modified Project 1 file has the name p1\_lib.c, then the header file must have the name p1\_lib.h. Both p1\_lib.c and the main Project 2 file must include the new header file (note double quotes):

#include “p1\_lib.h“

All files must be in the same directory, then you can compile two source code files together:

gcc p2\_main.c p1\_lib.c -o proj2

TASK #5. ELABORATING ON THE LINKED LIST NODES

Add a structure with IEEE 754 representation and a float field to your nodes. The new fields must be calculated before the node is added to the list. Use your Project 1 library to calculate the IEEE 754 representation for the random number.

Add a new function to your library that restores the decimal value from the IEEE 754 representation. Use this function to calculate and save the restored number in the field. If the restored number is not equal to the original number, the node should not be added but the error message should be printed to stderr.

**PROGRAMMING SUGGESTIONS**

* + Work on your project step-by-step. If you cannot fulfill all requirements but successfully solve some tasks you will be graded anyway. If your code does not compile by gcc, you will have a zero or very low grade.
  + Use a function for every logically separated piece of code.
  + Apply a correctness check for inputs and results.
  + **Do not copy any piece of code from online resources or another person. All works will be checked for plagiarism and academic integrity will be strongly enforced.**

**PROGRAM EXECUTION**

./proj2 100 dat

After running the above, the four files must be created. The examples are attached.

**SUBMISSION**

You must submit only one, **the most successful**, version. Please review your work carefully before submitting it. Also, check the grading rubric to ensure that your program meets all requirements.

Your lab instructor will provide you with a detailed description of the submission process.

**POLICY ON CHEATING**

**This assignment is to be completed individually – cheating is not tolerated**. We will be comparing your code against that of other students in the class and similar assignments from online platforms including Generative AI. All students involved in a cheating accident (i.e., whether sharing or receiving code) will be penalized. Please, read the syllabus for our policies on cheating and refer to *Programming Assignments Requirements and Recommendations* for examples of what we consider cheating, as those are not only limited to what you submit. Students caught cheating will receive 0 points for the assignment and will be reported.