CS 211 Homework 2

Jay Kania

Fall 2023

Introduction

In this assignment, you will implement an Advanced Bitwise FizzBuzz (see more about FizzBuzz here). This task involves applying FizzBuzz logic to each bit of a 32-bit integer, with a twist of handling the most significant bit (MSb) differently. All numbers will be 32-bit ints represented in 2's complement form. You will practice bitwise operations, string handling, and command-line argument processing in C.

```
./hw2 {32-bit integer}
```

void preComputeResults()

Purpose: To pre-compute and store the FizzBuzz results for each bit position in the cache array.
Tasks:

- For every possible bit position, fill in the logic to store 'Z' for positions divisible by both 3 and 5, 'R' for positions divisible by 3, and 'U' for positions divisible by 5. **You should only perform this replacement if the bit is set to 1.**
- Ensure that non-replaceable bits are handled appropriately (e.g., non-divisible bits and zero bits).

Example:

Bits	1	1	1	0	0	1	0	1
Pos	7	6	5	4	3	2	1	0
Out	**S**	**R**	**U**	**0**	0	**1**	0	**Z**

Observations:

1. Position 7 is set to S since the MSb = 1 (see more in AdvancedBitwiseFizzBuzz())
2. Position 6 is set to R since Bit = 1 and bit position 6 is divisible by 3
3. Position 3 is set to 0 since Bit = 0, even though bit position 3 is divisible by 3.

char* getFizzBuzzForBit(int position)

Purpose: To retrieve the pre-computed FizzBuzz result for a specific bit position.
Tasks:

- Implement the function to return the correct FizzBuzz value from the cache for the given bit position.

void advancedBitwiseFizzBuzz(int32_t N)

Purpose: To perform the advanced Bitwise FizzBuzz logic on the input number.
Tasks:

- Apply the FizzBuzz logic to each bit of the input number.
- Replace the sign bit with 'S' only if the number is negative.
- Print the bit or its FizzBuzz result, with a space every four bits.
- Ensure the bitstring is outputted from left to right, MSb to LSb.

Getting Started

You should first download and expand the provided files:

```
tar xvf hw2-provided.tar
```

You can download these locally and then copy them to ilab using scp: **scp hw2-provided.tar yourNetID@ilab1.cs.rutgers.edu:~/cs211** (assuming you have a cs211 directory in your root directory).

NOTE: ALL COMPILATION, TESTING, AND EXECUTION MUST BE PERFORMED ON THE ILAB. This is compulsory to ensure a standardized testing and grading environment for all students.

To compile the program, you should use gcc:

```
gcc hw2.c -o hw2
```

And to run it you should:

```
./hw2 {int32}
```

Autograder

Make sure your autograder is in the same directory as hw2.c.

Run the following command:

```
python3 hw2-autograder.py
```

Add test cases as needed on line 40 of hw2-autograder.py

Submission

ONLY SUBMIT hw2.c to Canvas. It is important that you only upload the completed C file. We will not accept a tar, zip, etc.