Assignment 2

Deadline 11:59PM Nov 11

Descriptions

The `sat` program performs a synthetic saturating addition operation. *Saturating* addition clamps the result into the representable range. Instead of overflowing with wraparound as ordinary two's-complement addition does, a saturating addition returns the type's maximum value when there would be positive overflow, and minimum when there would be negative overflow. Saturating arithmetic is a common feature in 3D graphics and digital signal processing applications.

Here are two sample runs of the sat program:

```
>>./a.out 8
8-bit signed integer range
min: -128   0xffffffffffffff80
max:  127   0x000000000000007f
>>./a.out 8 126 5
126 + 5 = 127
```

The program reports that an 8-bit signed value has a range of -128 to 127 and if you attempt to add 126 to 5, the result overflows and sticks at the maximum value of 127.

Your task is to implement the functions below to support saturating addition for the sat program.

```
long long signed_min(int bitwidth);
long long signed_max(int bitwidth);
long long sat_add(long long operand1, long long operand2, int bi
twidth);
```

The bitwidth argument is a number between 4 and 64. A two's-complement signed value with bitwidth total bits is limited to a fixed range. The signed_min and signed_max functions return the smallest and largest values of that range. The sat_add function implements a saturating addition operation which returns the sum of its operands if the sum is within range, or the appropriate min/max value when the result overflows. The type of the two operands is long long but you can assume the value of the operands will always be within the representable range for a bitwidth-sized signed value. That being said, this means there may be more bits than you need for a bitwidth-sized value, and these extra bits should be set appropriately to ensure the value is correctly interpreted when it is inside a long long.

Requirements

- **No relational operators or math.h functions**. You are prohibited from making any use of the relational operators. This means no use of < > <= >=. You may use != ==. You also should not call any function from the floating point math.h library (e.g no pow, no exp2). These restrictions are intended to guide you to implement the operation via bitwise manipulation. All other operators (arithmetic, logical, bitwise, ...) are fine.
- **No special cases based on bitwidth.** Whether the value of bitwidth is 4, 64, or something in between, your functions must use one unified code path to handle any/all values of bitwidth without special-case handling. You should not use if switch ?: to divide the code into different cases based on the value of bitwidth. This doesn't mean that you can't use conditional logic (such as to separately handle overflow or non-overflow cases), but conditionals that dispatch based on the value of bitwidth or make a special case out of one or more bitwidths are disallowed.
- **No loops.** No loops at all. (In particular, no loop increment by one and stop at max.)

A solution that violates any of these restrictions will receive **zero**, so please verify your approach is in compliance! Corner cases must be handled with a proper warning message and should exit the program.

Other additional submission requirements are:

1. The file you submit should be named: assignment2.c
2. Not following those will result in **zero**:
    1. Please comment your code
    2. Do not change function signatures including spaces in the skeleton code and what is provided in this handout
3. Run command will be the same as in above examples.

Grading

This assignment will be marked out of 8. For full marks this week, you must:

- (1 point) Correctly use git/GitHub and the repository following the handout
- (4 points) Generate a correct solution to the problem(s) in this handout
- (2 points) Handle corner cases
- (1 point) Reasonable comments that explain functions and variables

Submissions

- Github classroom link is posted on Learning Hub.
- assignment2.c (please do not capitalize “a”. All lower cases for the file name)
- AXXXX_AXXX.txt (empty file, but with your and your partner’s A number)
- Your code should compile and run without any additional files
- Only push these two files before the deadline.