Contributors: Kelvin Ihezue (ki120), Bryan Shangguan (bys8)

PART 2

To extract the top N bits from a 32 bit unsigned int, the most efficient method would be bitwise shift to the right. The function shifts the input value to the right by 32 - N positions. This operation moves the N most significant bits into the least significant positions of the int. The result is then returned.

To set a specific bit within the bitmap:

1. Find the Bit: First, find the correct byte within the char array by dividing the bit index by 8. The specific position of the bit within that byte is found by modding the index.

2. Apply a Mask: A mask is created by left shifting the value 1 by the bit_offset. The result is a byte that has a 1 only at the target position. The bitwise OR operator is then used to apply this mask to the byte.

PART 3

Explanation of the Bug:

The program gets stuck in an infinite loop because of how setcontext and uc_link work together. When main calls setcontext(&worker1_ctx), it jumps into the worker. After the worker finishes, it comes back to main through uc_link, but it resumes right where setcontext was called. That makes main set up the worker again and call setcontext again, so the same process keeps repeating and the program never ends.

Change to Ensure Termination:

To stop the loop, we add a guard variable (resumed_to_main). The first time main runs, we set this flag. Then, when the worker finishes and control comes back to main through uc_link, the flag tells us we've already been here once. Instead of setting up the worker again, main just frees the stack and ends the program. This way, the worker only runs once, comes back to main once, and the program finishes normally.

- Bug Illustration:

Main

↓ (init_context prepares worker1_ctx)
Main
↓ (setcontext -> jump to worker)
Worker1 runs
↓ (finishes, jumps back via uc_link to main_ctx)
Main resumes
↓ (but no guard, so it re-runs the same logic)
Main
↓ (init_context prepares worker1_ctx again)
Main
↓ (setcontext -> jump to worker again)
Worker1 runs
↓ (finishes, jumps back via uc_link)
... repeats forever ...

- Fixed Code:

Main
↓ (init_context prepares worker1_ctx)
Main
↓ (setcontext -> jump to worker)
Worker1 runs
↓ (finishes, jumps via uc_link)
Main resumes
↓ (guard detects return)
Main cleans up and exits ✅