Initially, I was so confused on why we should use bit vector on this assignment. I was wondering why can't we just do vector[i]=1 for bv_set_bit(). So I went to lab and people gave me answers like if we go vector[i]=1, we are only accessing the byte not bit. I was still confused on the answer. So, I did some research on YouTube, and I finally understand the need of using bit vector. Since we are having an 8 bits array, each element contains 8 bits. While we only need store a Boolean value, true or false for each number, if we do vector[i]=1, the element looks something like 00000001 bitwise. On the other hand, if we want to clear the bit as vector[i]=0, it becomes 00000000. Which wastes 7 bits since we have 7 leading zeros in both cases. Therefore, we can use bit mask to store up to 8 Boolean values for 8 numbers.

To determine what index the number belongs to, we can just divide the number by 8, or right shift its binary base by 3. After we find which byte the number belongs to, we can try to find which bit is storing the Boolean value of the number. I used 7, because 7== 00000111, as my bit mask and it will help us the find the bit by doing (number & 7), which also equals number%8.

To set a bit, I use the OR operator. You first shift a 1 to the desired bit, other bits remain 0. Then we do EX. original byte OR 00010000. It will set only the bit we want to change but not the others. Alternately, we can use the AND operator to clear a bit. We can do EX. original byte AND NOT(00010000), which equals to original byte AND 111011111. It will clear the bit. This method saves a lot of memory, EX. I store 1000 numbers in the array but we only used 142 bytes according the valgrind.

I realized my program could be run faster if I also use bit vector the store the Fibonacci, Lucas and Mersenne numbers. The way my program determines whether a number is a special prime is to calculate the sum if first two terms and see if they can add up to the number every time. If I use bit vector, I could simply run the calculation once and store them, then determine whether the number exist in the array.

Cited: https://www.youtube.com/watch?v=SYoJ6gUXZvc&t=87s
Copied sieve.c from lab