Problem 1. You are given a positive integer \( n \) (that is stored in a register of the CPU). Design an algorithm to determine whether \( n \) is an even number. Your algorithm should have a cost no more than 10.

Problem 2. You are given two positive integers \( n \) and \( m \) (stored in two registers of the CPU). Design an algorithm to calculate \( n \mod m \). Your algorithm should have a cost no more than 10.

Note: \( n \mod m \) is the “remainder” of \( n \) divided by \( m \). For example, \( 10 \mod 2 = 0 \) and \( 13 \mod 3 = 1 \).

Problem 3. You are given a positive integer \( n \) (that is stored in a register of the CPU). Design an algorithm to determine whether \( n \) is a prime number. Your algorithm should have a cost no more than \( 100\sqrt{n} \). Note that calculating \( \sqrt{n} \) is not an atomic operation.

Problem 4. You are given two positive integers \( n \) and \( m \) (stored in two registers of the CPU), where \( n \) is a power of 2. Design an algorithm to calculate \( m^n \). Your algorithm should have a cost no more than \( 100\log_2 n \).

Problem 5. You are given two sets \( S_1 \) and \( S_2 \) of integers. Specifically, \( |S_1| = n \) (that is, the number of integers in \( S_1 \)—the size of \( S_1 \)—is \( n \)) while \( |S_2| = m \). The integers in \( S_1 \) and \( S_2 \) have been stored in memory as shown in the figure below. In particular, the integers in \( S_1 \) have been sorted in ascending order, while those in \( S_2 \) have not. The starting address \( x \) of \( S_1 \) and the starting address \( y \) of \( S_2 \) have been stored in the CPU. So are the values of \( n \) and \( m \).

Design an algorithm to determine whether \( S_1 \cap S_2 \) is empty—in other words, whether the two sets have a common integer. Your algorithm should have a cost no more than \( 100m \log_2 n \).