The Setup:

You are given an infinite amount of N different types of balls - call them {a1,a2,a3,...aN}
You are also given a set of S trash cans each having a capacity of (w+x) where w and x are natural numbers.
There are two random independent functions - f1 and f2 which determine where each ball goes. That is, fi is a function from {a1,a2,...aN}-> {1,2,...S}. When a ball ai is thrown into the set of trash cans, the following occurs:
1] If a ball of type ai is currently in the set of trash cans, nothing happens and the newly thrown ball disappears (i.e. the set of trash cans will never have >= 2 balls of the same type), if it isnt there:
2] f1(ai) is computed and f2(ai) is computed
3] The Remaining capacity of trash cans f1(ai) and f2(ai) are compared, the remaining capacity of a trash can is defined as (w+x) - (no. of balls currently in the trash can)
4] There are now a few cases
4.a] Case 1 : f1(ai) != f2(ai) : In this case, the ball goes into the trash can having larger remaining capacity.
4.b] Case 2 : f1(ai) = f2(ai) != 0 : In this case, the ball goes into either f1(ai) or f2(ai) randomly.
4.c] Case 3 : f1(ai) = f2(ai) = 0 : The system crashes and the game is over
5] Now, if on throwing this ball, the net number of balls in the entire set of trash cans exceeds S*w, then a random ball is chosen from any trash can and is removed (i.e. the total number of balls in the set of all trash cans can never exceed S*w)
f1 and f2 remain constant throughout the system's lifetime.
Assume w = 16, x = 12 and N is of the order of 2^34, S can be assumed to always be a power of 2 and around 16384
It is given that 0 < x < w always

Part A:

An attacker interacts with the system with intention of crashing it and ending the game. However he does not have access to f1 or f2, nor does he have access to the set of S trash cans. He can also reset the system to having 0 balls. The only way he can interact with the system is by throwing balls, the system gives him the following information:
1] Whether the ball he threw was in the system or not
2] Whether his ball throw caused the system to crash
There is an O(S^(2*(w+x))) algorithm to guarantee this, using the pigeonhole principle, it works by creating an initial set of (w+x)*S^2 + 1 balls, and then checking all (2*(w+x) + 1) - tuples of the initial set. For the given values of w and x, this method is far slower than simply throwing random balls and hoping probability helps you. Either prove that the attacker cannot do better than probabilistically throwing balls OR find a better algorithm/method.

Part B:

The attacker has found a way to undermine step 5 of what happens when a ball is thrown! Now when the total number of balls in the entire set of trash cans exceeds S*w, the system will prompt the attacker and ask him which ball should be removed. Note the attacker still doesn't know f1(ai) or f2(ai) for any ai, the system simply gives him the list of S*w + 1 balls in the set of trash cans and asks him to choose one to remove. Can the attacker now do better? If so, how?