1. Let \( n_1 \) be the number (out of the total \( n \)) of steps during which the elephant moved left and let \( n_2 \) be the number of steps during which the elephant moved right. The distance between the elephant and the origin after \( n \) steps is \( |n_1 - n_2| \).

Using Chernoff bounds, the value of \( n_1 \) is no more than \( n + c_1 \alpha \sqrt{n \log n} \) and no less than \( n - c_2 \alpha \sqrt{n \log n} \) with probability \( \geq 1 - n^{-\alpha} \), for some constants \( c_1 \) and \( c_2 \). The same can be said about the value of \( n_2 \) as well.

As a result it follows that \( |n_1 - n_2| \) can be no more than \( c \alpha \sqrt{n \log n} \) with probability \( \geq 1 - n^{-\alpha} \), for some constant \( c \).

2. Let us calculate the expected number of Hamiltonian paths in a random tournament \( T(V,E) \) where every edge of \( G(V,E) \) has a random orientation, chosen independently with probability \( \frac{1}{2} \).

Let \( \pi \) be a random permutation of \( \{1, \ldots, n\} \). What is the probability that the sequence of nodes \( \pi(1), \pi(2), \ldots, \pi(n) \) corresponds to a Hamiltonian path? This sequence will be Hamiltonian path if \( (\pi(i), \pi(i + 1)) \) is a directed edge in the tournament, for \( 1 \leq i \leq (n - 1) \). The probability of this is \( \frac{1}{2^n} \). As a result, it follows that the expected number of Hamiltonian paths in a random tournament is \( \frac{n!}{2^n} \). Therefore, there exists a tournament that has at least \( \frac{n!}{2^n} \) Hamiltonian paths.

3. Here we use the fact that two sorted sequences of length \( N \) can be merged in \( O(\log \log N) \) time using \( N \) CREW PRAM processors. We recursively merge the sequences \( X_1, X_2, \ldots, X_{k/2} \) to get \( Y_1 \) using \( \sum_{i=1}^{k/2} |X_i| \) processors. At the same time we recursively merge \( X_{(k/2)+1}, X_{(k/2)+2}, \ldots, X_k \) to get \( Y_2 \) using \( \sum_{i=(k/2)+1}^{k} |X_i| \) processors. The total number of processors used is clearly \( n \).

Followed by the above recursive steps, we merge \( Y_1 \) and \( Y_2 \) using \( n \) processors. This merging will take \( O(\log \log n) \) time. Let \( T(k) \) be the time needed to merge \( k \) sorted sequences using \( n \) processors. Then, we have:

\[
T(k) = T(k/2) + O(\log \log n).
\]

The above recurrence relation can be solved to get: \( T(k) = O(\log k \log \log n) \).

4. In this problem we use the fact that we can sort a sequence of \( n \) integers in the range \([1, n \log^c n]\) in \( \tilde{O}(\log n) \) time using \( \frac{n}{\log n} \) Arbitrary CRCW PRAM processors, where \( c \) is any constant.

Let \( X_1, X_2, \ldots, X_k \) be the input sequences. We start by sorting each of these sequences using the above algorithm. This will take \( \tilde{O}(\log n) \) time using \( \frac{nk}{\log n} \) processors. Consider a complete binary tree where each leaf has one of these sorted sequences. We traverse this tree level by level starting at the level immediately above the leaves. At any node in the tree we compute the intersection of the two children.
We can compute the intersection of two sorted sequences of length at most $q$ each (where $q \leq n$) using $\frac{q}{\log n}$ processors in $\tilde{O}(\log n)$ time. This means that the computations in each level of the tree can be computed in $\tilde{O}(\log n)$ time using $\frac{nk}{\log n}$ processors.

As a result, the entire algorithm runs in time $\tilde{O}(\log n \log k)$ using $\frac{nk}{\log n}$ processors.

**Extra Credit:** There are two cases to consider.

**Case 1:** $k \leq \log n$: In this case use the previous algorithm.

**Case 2:** $k > \log n$: Note that the number of nodes in any level of the tree is one half of the number of nodes in the level below. We use the above technique of computing intersections for the bottom $2 \log \log n$ levels. The number of nodes in the $2 \log \log n$th level from the bottom is $\frac{n}{\log \log n}$. From this level on, use the $\tilde{O}\left(\frac{\log n}{\log \log n}\right)$ time algorithm for integer sorting (and hence intersections). The total time is $\tilde{O}\left(\log n \log \log n + \frac{\log n}{\log \log n} \log k\right)$. Number of processors needed at any level of the tree is no more than $\frac{nk}{\log n}$. 