1. A basic and often useful data structure is a doubly linked list. In internal memory it uses $O(N)$ space and supports the following operations: Given a pointer to an element x in the list, a new element can be inserted before or after x in $O(1)$ time, and x can be deleted in $O(1)$ time. Similarly, given a pointer to x, the K-element sublist starting with x can be traversed in $O(K)$ time, for any $K \geqslant 1$. Here your goal is to construct an I/O-efficient doubly-linked list. The structure should occupy $O(N/B)$ blocks, should support insertions and deletions as above using $O(1)$ I/Os, and should allow the traversal of any K-element sublist using $O(1 + K/B)$ I/Os. Argue that the structure you develop achieves these space and I/O bounds.

2. Let $X[0..n-1]$ and $Y[0..n-1]$ be two arrays, each storing a set of n numbers. Let $Z[0..n-1]$ be another array, in which each entry $Z[i]$ has three fields: $Z[i].x$, $Z[i].y$ and $Z[i].sum$. The fields $Z[i].x$ and $Z[i].y$ contain integers in the range $\{0, ..., n-1\}$; the fields $Z[i].sum$ are initially empty. We wish to store in each field $Z[i].sum$ the value $X[Z[i].x] + Y[Z[i].y]$. A simple algorithm for this is as follows.

   ComputeSums(X, Y, Z)
   1: for $i = 0$ to $n-1$ do
   2: $\quad Z[i].sum = X[Z[i].x] + Y[Z[i].y]$
   3: end for

   (i) Analyze the number of I/Os performed by ComputeSums.

   (ii) Give an algorithm to compute all $Z[i].sum$ that performs only $O(Sort(n))$ I/Os.

3. Prove that in Van Emde boas layout of n numbers, searching a number needs at most $4\log_B(N/B)$ I/Os. If we search all numbers existing in the tree in an arbitrary order, prove a search needs $2\log_B(N/B)$ I/Os in average.

4. An N-element B-tree occupies $O(N/B)$ disk blocks and supports insertions, deletions, and FIND operations using $O(\log_B N)$ I/Os. Here your goal is to use the buffering idea to obtain a structure with the same space requirements and the same cost for FIND operations, but which uses only $O((1/\sqrt{B})\log_B N)$ amortized I/Os per insertion or deletion. Argue that your structure achieves these space and I/O bounds. For simplicity, you can assume there is not any deletion.

5. Batched range reporting is the following problem: Given a set R of rectangles in the plane and a set P of points in the plane, report all pairs (p, r) such that $p \in P, r \in R$, and $p \in r$. If $N = |P| + |R|$ and K is the number of such pairs (p, r) reported, your task is to design an algorithm that solves this problem using $O(sort(N) + K/B)$ I/Os. Provide a description, analysis, and proof of correctness of your algorithm.