Clustering and nonclustering indexes. In a clustering index we assume that all tuples having the same search key value are stored in at most \( O(c) \) blocks, where \( c \) is the minimum number of blocks required to store them. In a non-clustering index only pointers to tuples are stored in the index structure (e.g., in the leaves of the B-tree). Denote by \( B \) the number of tuples that fit in one block.

1. We want to do a select operation \( \sigma_{a_i=374}(R) \), where \( a_i \) is an attribute of \( R \). Let \( t \) be the number of tuples in the result (i.e., \( t = |\sigma_{a_i=374}(R)| \)).

   Consider four types of indexes for \( R \) on \( a_i \):
   
   (a) Hash index, which is a clustering index.
   (b) Hash index, not clustered.
   (c) B-tree index, which is a clustering index.
   (d) B-tree index, not clustered.

   Denote by \( u \) the number of tuples of \( R \) that hash to the same bucket as 374, and by \( n \) the degree of the B-tree.

   What is the worst case I/O-complexity (as a function of \( B \), \( |R| \), \( t \), \( u \), and \( n \)) of select using each of the four types of indexes?

2. We want to do join, \( R(X,Y) \bowtie S(Y,Z) \), where \( Y \) is a primary key. There is a hash index on \( Y \) for \( R \) (clustering index). The number of buckets is much larger than the number of blocks in main memory and one block suffices to store each bucket. There is a B-tree index on \( Y \) for \( S \) (also a clustering index).

   Consider three algorithms for join:
   
   (a) Scan \( S \): for each tuple \( s \in S \) look up the join-value for \( s \) in the index of \( R \).
   (b) Sort \( R \) and merge the two relations (as in the sorting based algorithms) using the sorted order in the B-tree index of \( S \).
   (c) Scan \( R \): for each tuple \( r \in R \) look up the join-value for \( r \) in the index of \( S \).

   For which sizes of \( R \), \( S \), and the main memory \( M \) is one of (a), (b), or (c) considerably worse or better than the others (I/O-complexity)? (Assume that \( R \) and \( S \) are both too large to fit in main memory.)