Name:

- Consider the query: `select * from R where 100 < a < 200 and b = 20;`
  Assume $|R| = 10,000,000$, and $b_R$ (number of blocks of $R$) = 100,000. So number of tuples in each block = 100. There is a primary B+-tree on $a$ (so $R$ is sorted by $a$), and a secondary B+-tree on $b$, both with height = 3. Number of tuples with $100 < a < 200 = 100,000$; Number of tuples with $(b = 20) = 50$; Number of tuples that satisfy both conditions = 5. Assume a leaf of either index contains 100 ptrs.

The following shows the steps in executing this query. For each of the steps, estimate the number of seeks and blocks transferred.

**Option 1:** Use the B+-tree index on $a$ to find tuples that satisfy the first condition, and for each matching tuple, check if $b = 20$.

- Starting at the root, search down till you find the leaf block with the first $a > 100$.
  
  3 blocks read + 3 seeks

- Follow the first pointer to a tuple with $a > 100$, and then do a sequential scan till you hit the first tuple with $a >= 200$.

**Option 2:** Use the B+-tree index on $b$, and for each matching tuple, check if $100 < a < 200$.

- Starting at the root, search down till you find the leaf block with the first $b = 20$.

- Since this is a secondary index: read through the leaf blocks until you find the first key $> 20$.

- For each of the pointers encountered during the above search, follow it to the relation blocks and return the corresponding tuple.

- Consider a relation $R$ with $b_R = 10000$ blocks. Compute the number of seeks and the total number of blocks transferred to sort $R$ and write the sorted output to disk. Assume memory = 101 blocks. Feel free to assume you have $\pm 2$ blocks of memory if that simplifies the calculations.

  1. Read first 100 blocks of $R$, sort and write to disk (**2 seeks, 200 blocks transferred**).
  2. Repeat the above step 99 more times to create 100 sorted runs on disk.

  (3) Do a 100-way merge, using 1 block each for the 100 runs and 1 block for the output.
Consider two relations $R$ and $S$ with the following characteristics: $|R| = 100,000$; $|S| = 1,000,000$; $b_R = 1000 \text{ blocks}$; $b_S = 5000 \text{ blocks}$. We want to do a join over the two relations. For each of the following scenarios, estimate the number of seeks and the total number of blocks transferred. Feel free to approximate a little bit if you like, especially if having one or two extra blocks of memory would help simplify the computations. For instance, for nested loops with 1002 blocks of memory, we have:

1. **Memory = 202 blocks, block nested loop join with $R$ as the inner.**
   
   We will give 200 blocks (of memory) to $R$, and 1 block to $S$ and 1 block to output
   
   (1) Read first 200 blocks of $R$ in memory.  
       1 seek, 200 blocks read
   
   (2) Scan $S$ and join with those 200 blocks of $R$.
   
   (3) Repeat 5 times.

2. **Memory = 502 blocks, hash join with $R$ as the inner.**
   
   Since $b_R > 502$, we have to partition both $R$ and $S$ into 2 partitions each.
   
   (1) Read and partition $R$ with $b_b = 167$ blocks (i.e., 167 blocks for $R$ and  
       for each of the partitions). So: we read 167 blocks of $R$, hash them into  
       appropriate partitions and repeat  
       2000 blocks transferred, $2000/167 = 12$ seeks
   
   (2) Read and partition $S$ with $b_b = 167$ blocks.
   
   (3) Read first partition of $R$, and build a hash table on it.
   
   (4) Scan $S$ and do the join.
   
   (5) Repeat once.