CS186 Vitamin #1

* Required

External Hashing and Sorting

Your "dream machine" allocates 32 MB for the buffer (memory) for its external hashing and sorting algorithms.

All input is read from disk, and all output is written to disk. The I/O cost is the number of page reads/writes, where each page is 128 KB large.

Note that 1024 KB = 1 MB.

1. **Q1: How many passes are required to fully sort a 8192 MB file with external merge sort?** *

2. **Q2: What's the I/O cost of fully sorting a 8192 MB file with external merge sort?** *

3. **Q3: Suppose we double the size of our buffer, to 64 MB. What is the largest file size (in MB) that we can externally sort in two passes?** *

4. **Q4: Generalizing Q3, if we double the size of our buffer, approximately how much larger of a file can we externally sort in k passes?** *

   Mark only one oval.

   ○ 2 times larger

   ○ 2^2 times larger

   ○ 2k times larger

   ○ 2^k times larger

   ○ k^2 times larger

For Q5 and Q6, use 32 MB for the size of the buffer.

5. **Q5: You decide to separate your 8192 MB dataset with external hashing. How does the I/O cost of externally hashing the file compare with the I/O cost of externally merge sorting the file?** *

   Assume that the data is uniformly distributed on the hashed key and that your hashing function distributes the records into partitions evenly.
   Mark only one oval.

   ○ External merge sort will use fewer I/O's

   ○ They have the same I/O cost

   ○ External hashing will use fewer I/O's

6. **Q6: Suppose you are hashing a file and one of the partitions is 36 MB after the first pass (all other partitions can fit in the 32 MB buffer). How much larger (in I/Os) is the cost of externally hashing this file, compared to a scenario (with the same file) in which no partitions are ever oversized?** *

   Assume that a new hash function is chosen for the second pass such that the records are distributed in a way that guarantees subsequent partitions to be under 32 MB.

Basic SQL Queries

Assume there exists a table called "Songs" with the following columns.

song_id (Int, Primary Key), artist_name (Text), title (Text), year_released (Int), length_seconds (Int), rating (Float)

An example record could look like the following:
(1, 'D.O.D.', 'Crazy Concurrency', 2007, 188, 10.0)

7. **Q7: Which SQL query (or queries) will get the number of songs released after 2010 with a rating of at least 9.0?** *

   There can be more than one correct answer. At least one answer is correct.
   Check all that apply.

   □ SELECT COUNT(*) FROM Songs WHERE year_released > 2010 AND rating >= 9.0;

   □ SELECT COUNT(*) FROM Songs GROUP BY year_released, rating HAVING year_released > 2010 AND rating >= 9.0;

   □ SELECT COUNT(*) FROM Songs WHERE rating >= 9.0 GROUP BY year_released HAVING year_released > 2010;

   □ SELECT COUNT(song_id) FROM Songs WHERE year_released > 2010 AND rating >= 9.0;

8. **Q8: Which SQL query (or queries) will get the list of artists, without duplicates, who have produced at least one song more than 5 minutes long? ***

There can be more than one correct answer. At least one answer is correct.
Check all that apply.

□ SELECT DISTINCT artist_name FROM Songs WHERE length_seconds > 300;

□ SELECT artist_name FROM Songs WHERE length_seconds > 300 GROUP BY artist_name;

□ SELECT artist_name FROM Songs WHERE length_seconds > 300 GROUP BY artist_name, length_seconds HAVING COUNT(*) >= 1;

□ SELECT artist_name FROM Songs GROUP BY artist_name, length_seconds HAVING length_seconds > 300;