1. Consider having relation R with 10000 records and there are 100 records per page and relation S has 2000 records with 50 records per page.

Calculate the following and explain your calculation:

- Number of pages for relation R:
- Number of pages for relation S:
- I/O cost for grace hash join between R and S:
  - Buffer pages = 12
  - Buffer pages = 6

- I/O cost for simple hash join between R and S:
  - Buffer pages = 23
  - Buffer pages = 13

2. When is simple hash join better to use compared to grace hash join and vice versa.

3. Explain the idea behind hybrid hash join and why it was proposed.
Solution
1. Consider having relation R with 10000 records and there are 100 records per page and relation S has 2000 records with 50 records per page. (assuming data is not skewed)

Calculate the following and explain your calculation:
- Number of pages for relation R: \( \frac{10000}{100} = 100 \).
- Number of pages for relation S: \( \frac{2000}{50} = 40 \).
- I/O cost for grace hash join between R and S:
  - Buffer pages = 12
    Smaller relation is S and we can see that \( \sqrt{40} < 10 \) that means we only need to partition the data once.
    Reading S and R + writing partitions of S and R + reading of partitions of S and R
    \( (|R| + |S|) + (|R| + |S|) + (|R| + |S|) \)
    \( 3 \times (100 + 40) = 420 \)
  - Buffer pages = 6
    Smaller relation is S and we can see that \( \sqrt{40} > 4 \) that means we will have 5 partitions with approximately 8 pages long and need to partition each partition again. Therefore, it will an extra read for R and S and an extra write.
    Reading S and R + writing partitions of S and R + reading of partitions of S and R
    + writing and reading of resulted partitions of S and R
    \( (|R| + |S|) + (|R| + |S|) + (|R| + |S|) + (|R| + |S|) + (|R| + |S|) \)
    \( 5 \times (140) = 700 \)
- I/O cost for simple hash join between R and S:
  - Buffer pages = 23
    Smaller relation is S and we can see that we can load half of relation S and build a hash table in memory. That means we will scan relation R and probe the in memory hash table.
    Reading S and R + writing and reading half of S and half of R
    \( (|R| + |S|) + 2 \times (|.5R| + |.5S|) \)
    \( (100 + 40) + 2 \times (50 + 20) = 280 \)
  - Buffer pages = 13
    Smaller relation is S and we can see that we can load 10 pages of relation S each iteration and build a hash table in memory. That means we will scan relation R and probe the in memory hash table for each iteration. Assuming data is not skewed, we can approximate the proportion of pages written to disk for S and apply the same for R. If data is skewed we can't guarantee that.
    Reading S and R + writing and reading of 30 pages of S and 75 pages of R + writing and reading of 20 pages of S and 50 pages of R + writing and reading of 10 pages of S and 25 pages of R
    \( (|R| + |S|) + 2 \times (|.75R| + |.75S|) + 2 \times (|.5R| + |.5S|) + 2 \times (|.25R| + |.25S|) \)
    \( (100 + 40) + 2 \times (75 + 30) + 2 \times (50 + 20) + 2 \times (25 + 10) = 560 \)
2. When is simple hash join better to use compared to grace hash join and vice versa.

   Simple hash join is better when most of the smaller relation can fit in memory and can finish join in 2 phases.

   Grace hash join is better when both relations are large compared to available buffer pages. (Big data)

3. Explain the idea behind hybrid hash join and why it was proposed.

   The idea was proposed to maximise the utilization of available memory where part of memory is used to build in memory hash table of partitions for the smaller relation and uses the rest of buffer pages, say k, as output buffers for the remaining partitions. After that we scan the larger relation where we run a hash function to check if tuple is in memory hash table. If it falls in a partition that is loaded in memory, another hash function h2, where h2 <> h1, is used to join it with matching tuple and is written directly to output result buffer. If it falls into remaining partitions, it sends it to the partition that it falls into using the output buffer for that partition. After that we perform grace hash join on the remaining partitions that was pushed to disk.