Here are two similar exercise sheets with adjusted numbers:

Implementation of DBMS
like Exercise Sheet 6

1) Swizzling Pointers

Suppose that if we swizzle all pointers automatically, we can perform the swizzling in half the time it would take to swizzle each one separately. If the probability that a pointer in main memory will be followed at least once is *p*, for what values of *p* is it more efficient to swizzle automatically than on demand?

---

2) Data File and Index Block Calculations

Suppose blocks hold either **15 records**, or **75 key-pointer pairs**. We have a data file with **1,500,000 records**. In this task, we assume that each block will be as full as possible.

Questions:
a) How many blocks do we need for the data file?
b) How many blocks do we need for a dense index?
c) How many blocks do we need for a sparse index?
d) We want to add higher-level index structures to the index considered in **b**. Describe how many blocks we have in these higher levels until we end up in a level with just one block.
e) Repeat **d** for the sparse index in **c**.

---

3) Disk I/O Calculations

Use the same scenario as in task 2. We assume that nothing is in memory initially, and that the search key is the primary key for the records. What is the average number of disk I/O's needed to retrieve a particular record that is stored in our data file given its search key using the following approaches?

Questions:
a) We do not use any index but sequentially inspect the data file from the beginning until we find the record.
b) We just use the dense index described in task **2b** to retrieve the record. We sequentially scan the index from the beginning until we find the search key value we are looking for.
c) We use the multi-level index described in task **2d** to retrieve the record.
d) We use the multi-level index described in task **2e** to retrieve the record.

Implementation of DBMS
Exercise Sheet 7

1) Swizzling Pointers

Suppose that if we swizzle all pointers automatically, we can perform the swizzling in **one-third** of the time it would take to swizzle each one separately. If the probability that a pointer in main memory will be followed at least once is p, for what values of p is it more efficient to swizzle automatically than on demand?

---

2) Data File and Index Block Calculations

Suppose blocks hold either **12 records**, or **60 key-pointer pairs**. We have a data file with **2,400,000 records**. In this task, we assume that each block will be as full as possible.

Questions:

a) How many blocks do we need for the data file?
b) How many blocks do we need for a dense index?
c) How many blocks do we need for a sparse index?
d) We want to add higher-level index structures to the index considered in **b**. Describe how many blocks we have in these higher levels until we end up in a level with just one block.
e) Repeat **d** for the sparse index in **c**.

---

3) Disk I/O Calculations

Use the same scenario as in task 2. We assume that nothing is in memory initially, and that the search key is the primary key for the records. What is the average number of disk I/O's needed to retrieve a particular record that is stored in our data file given its search key using the following approaches?

Questions:

a) We do not use any index but sequentially inspect the data file from the beginning until we find the record.
b) We just use the dense index described in task **2b** to retrieve the record. We sequentially scan the index from the beginning until we find the search key value we are looking for.
c) We use the multi-level index described in task **2d** to retrieve the record.
d) We use the multi-level index described in task **2e** to retrieve the record.

Implementation of DBMS
Exercise Sheet 5

1. Suppose that if we swizzle all pointers automatically, we can perform the swizzling in one-third of the time it would take to swizzle each one separately. If the probability that a pointer in main memory will be followed at least once is q, for what values of q is it more efficient to swizzle automatically than on demand?

2. Suppose blocks hold either 12 records, or 40 key-pointer pairs. We have a data file with 2×10^5 records. In this task, we assume that each block will be as full as possible.
   a) How many blocks do we need for the data file?
   b) How many blocks do we need for a dense index?
   c) How many blocks do we need for a sparse index?
   d) We want to add higher-level index structures to the index considered in (b). Describe how many blocks we have in these higher levels until we end up in a level with just one block.
   e) Repeat (d) for the sparse index in (c).

3. Use the same scenario as in task 2. We assume that nothing is in memory initially and that the search key is the primary key for the records. What is the average number of disk I/Os needed to retrieve a particular record that is stored in our data file given its search key using the following approaches?
   a) We do not use any index but sequentially inspect the data file from the beginning until we find the record.
   b) We just use the dense index described in 2b to retrieve the record. We sequentially scan the index from the beginning until we find the search key value we are looking for.
   c) We use the multi-level index described in 2d to retrieve the record.
   d) We use the multi-level index described in 2e to retrieve the record.

Here are similar **practice questions** designed to deepen your understanding of database concepts, inspired by the sample provided:

Implementation of DBMS

same to Exercise Sheet 5

---

1. Pointer Swizzling and Memory Efficiency

1. Suppose pointer swizzling requires a fixed overhead of 20% more memory for bookkeeping. If the probability of following a pointer at least once is p, what values of p make it more memory-efficient to swizzle pointers on-demand rather than preloading them?
2. If swizzling speeds up pointer dereferencing by 30%, for what values of ppp is it beneficial to swizzle automatically despite the overhead?

---

2. Block Utilization and Indexing

1. Suppose blocks can hold 100 key-pointer pairs or 25 records, and the data file contains 10^5 records. Assume all blocks are as full as possible.
   a) How many blocks are needed for the data file?
   b) How many blocks are required for a dense index?
   c) How many blocks are required for a sparse index if each block of the data file corresponds to one index entry?

   d) We want to add higher level index structures to the index considered in b). Describe how many blocks we have in these higher levels until we end up in a level with just one block.

   e) Repeat d) for the sparse index in c).

---

3. Disk I/O Analysis for Different Search Strategies

1. Consider a data file with 10^6 records and a block size of 4096 bytes, where each record is 128 bytes. Assume a dense index is available, and the primary key is used for searching.
   a) How many disk I/Os are required on average if there is no index, and the file is searched sequentially?
   b) What is the average number of disk I/Os if only a single-level dense index is used for searching?
   c) Extend the dense index to a multi-level index. How many I/Os are required to retrieve a record, assuming the top-level index is always in memory?
   d) Compare the I/Os for a B+ tree index with a fan-out of 100 for the same dataset.

4. Record Alignment and Packing Efficiency

1. A record consists of the following fields:
    - A fixed-length string (20 bytes)
    - An integer (4 bytes)
    - A floating-point number (8 bytes)
    - A variable-length string with an average size of 15 bytes.

a) Calculate the size of the record if fields are stored sequentially with no alignment.
b) If fields are aligned on 4-byte boundaries, what is the total size of the record?
c) How many records can fit in a block of size 4096 bytes, assuming the block has a 64-byte header?
d) If variable-length fields are stored separately with a 4-byte pointer in the record, how does this impact the number of records per block?

---

5. Average Record Size with Variable-Length Fields

1. A customer record has the following structure:
    - Customer ID: 10 bytes (fixed length)
    - Name: Variable length (10-50 bytes, uniformly distributed)
    - Address: Variable length (30-100 bytes, uniformly distributed)
    - Notes: Variable length (0-500 bytes, uniformly distributed).

Assume each record includes a 4-byte pointer for each variable-length field and an 8-byte header for metadata.
a) Calculate the average size of a customer record.
b) How many customer records can fit into a block of size 8192 bytes?

---

6. Access Pattern Optimization

1. Consider a database storing product records. Blocks hold 50 records each, and there are 2*10^5 records in total.
   a) If the search key is uniformly distributed, what is the expected number of block accesses to find a record in a sequential scan?
   b) How does this change if a sparse index with one index entry per block is used?
   c) Compare the performance of a two-level index structure versus a B+ tree with a fan-out of 75 for searching.