Aleksas Murauskas 260718389
Fouad Bitar 260719196
Ege Odaci 260722818

1. Provide all major data structures you would use in the file system design. Categorize them as: on-disk and in-memory. Note that all on-disk data structures need to deal with the block-oriented storage organization. Suppose a directory entry is 100 bytes (for discussion sake), a 1024-byte disk block would hold 10 directory entries plus a portion of the 11th block.
    a. In-Memory Structures: Open file descriptor table, cached I nodes, File Descriptor Tables
    On-Disk Structures: super block, I nodes, the root directory, data blocks, free block list

2. Categorize the in-memory data structures as primary data holders and cache memory data holders. You obtain better performance from the file system by increasing the number of cache memory data holders in the system. However, you need to allocate primary data holders for a properly functioning file system.
    a. Primary data holders: Open File Descriptor Table
    Cache memory data holders: Cached I nodes

3. Provide the pseudo code for a function that initializes the simple file system. Your simple file system is running over a disk (an array of blocks that are randomly addressable). The initialization function is responsible for writing the formatting information onto the disk. You cannot use the disk without the initialization step. That is, you cannot create a file before running this step. The initialization step does not store any data. It would create the root directory and make the file system ready for storing files.
    I. Allocate memory to act as a disk containing, a Super block, several I nodes and Data Blocks
    II. Initialize a superblock
    III. Create and Initialize a root directory
    IV. Set I node 1 to point to the root directory
    V. Initialize Open file descriptor table
    VI. Initialize File Descriptor Tables

4. Provide the pseudo code for a function that creates a file. In this file system we don't have any directories. So, all files are stored in the root directory of the file system.
    I. Check if a file by this name already exists
    II. Check if that file is already open
    III. Check if an inode is available for the file
    IV. If all checks are validated:
    V. Initialize I node
    VI. Set I node attributes, size is 0

VII. Update directory entry with mapping
VIII. Update open file descriptor table if necessary

5. Provide the pseudo code for a function that writes to a file. Assume that the file is already open.
    I. Find the Open File Descriptor Table in the File Descriptor table
    II. Allocate an appropriate number of blocks to fit the new file size
    III. Modify the pointers in the I node in order to point to the newly allocated

6. Provide the pseudo code for a function that reads from a file. Assume that the file is already open
    I. Find the file's file descriptor in the Open File Descriptor Table
    II. Obtain the read pointer from the file descriptor
    III. Read the number of Bytes in the file for the desired location
    IV. Use the pointer and the number of bytes obtained to read from the file

7. Provide the pseudo code for a function that closes the file system.
    I. Send all modifications not yet made to the disk
    II. Free Directory Table
    III. Free Open File Descriptor table
    IV. Free Cached I Tables
    V. Free File Descriptor Tables
    VI. Close Virtual Disk