Exercise Problems on lseek() and dup()

lseek Practice Problems:

1. **File Positioning:** Write a C program that opens a file, moves the file pointer to the end of the file using the `lseek` system call, and then reads and prints the last 100 bytes of the file.

2. **Random Access:** Create a binary file with some data records. Write a program that allows the user to input a record number and then uses `lseek` to move to that record and display its contents.

3. **File Truncation:** Write a C program that truncates a file to a specified size. Prompt the user for the new size and then use the `lseek` and `ftruncate` system calls to achieve this.

4. **Copying Files with `lseek`:** Implement a program that copies the contents of one file to another using the `lseek` system call to navigate through the source file. Ensure that it can handle files of different sizes.

dup Practice Problems:

1. **Redirecting Output:** Write a C program that forks a child process. In the child process, use the `dup` system call to redirect `stdout` to a file.
Demonstrate this by having the child process print something to `stdout`, which should be written to the file instead of the console.

2. **Piping with `dup`:** Create a program that forks two child processes. The first child writes data to `stdout`, and the second child reads data from `stdin`. Use the `dup` system call to create a pipe between the two processes, allowing the first child's output to be sent as input to the second child.

3. **File Descriptor Manipulation:** Write a C program that opens a file, duplicates the file descriptor using the `dup` system call, and then closes both the original and duplicated file descriptors. Verify that you can still read from the duplicated descriptor.

4. **Redirecting Input:** Build a program that redirects the standard input (`stdin`) to read from a file using the `dup2` system call. After redirection, prompt the user to enter data, which should be read from the specified file instead of the keyboard.