man ls | grep -w -m1 -- "-S" > output.txt

In Linux and other Unix-like operating systems, the **ls** command is frequently used. It is used to list the contents of a directory and stands for "list". The names of the files and subdirectories in the current directory are displayed when the **ls** command is used without any parameters.
-w: The grep -w option stands for "word-regexp." It instructs grep to look for entire words only, avoiding partial matches.
-m1: The grep -m option stands for "max count." It tells grep to terminate after the first match is discovered, rather than searching the full input.
- -: The double hyphen -- is commonly used in command-line parameters to indicate the conclusion of an options list. Anything beyond that is considered an argument, not an option. It ensures that -S is interpreted as an argument rather than an option in this circumstance.
"-S": This is the pattern that **grep** searches for in the manual page. In this case, it's the string "S". The double quotes are used to specify the exact string to search for, ensuring that any special characters or hyphens are interpreted correctly. The -S option in the ls command is used to sort the listed files and directories in descending order by file size. This means that the largest files will be presented first, followed by smaller files in decreasing order. When you wish to quickly find the largest files or folders within a directory, the -S option comes in handy.
I picked **ls** command and **-S** option because they work well together to swiftly find and manage huge files and folders within a directory. The -S option, which sorts files and directories by size in descending order, enables me to quickly identify the largest files or directories, making it a useful tool for disk space analysis and management.

My program is 2a. They are siblings because they are both child processes of the same parent process. They can run concurrently because they are independent processes, and the use of pipes (`pipe(fd)` and `pipe(fd2)`) allows them to communicate and exchange data while running simultaneously. The parent process can wait for their completion using `wait(NULL)` without blocking their concurrent execution.