The chosen command is: man rm | grep "^ *-v" -w -A 1 > output.txt
"rm" command removes files or directories and if we use this command with -v option, it will explain what is being done in the console. I have chosen "rm" command because I think that removing a file is one of the most important operations in a computer. The reason I have chosen "-v" as a option is that I really like to learn what happened after a command. For instance, when "rm" command executed without any option, it removes the file but does not give any notification about it. However, when it used with "-v" option, it tells that "removed [filename]". Therefore, it prevents any confusion for user.
Why I have chosen these extraOptions ("^ *-v" and -w -A 1):
Firstly, I tried to do my main command as: "man rm | grep "\-v" > output.txt". However, when we do it like that it founds every line that consist of "-v" not only the option part. For instance, if there is a "-v" in description part, it will also show that as a result.
Therefore, I modified it and resulted with upper main command.
"-w" is a option of grep to just match the whole word in the sentence. Therefore, it will not match --version as a result.
"-A 1" is a option of grep to print 1 line of trailing context after matching lines. Therefore, we can also take the description of the option.
"^" allows us to look only for beginning of lines.
"*" matches up zero or more times the preceding character (definition taken from google).
At the end, we are able to find the our option and its definition.

My program:
Our main process is "Shell process" with its own unique PID. Then, I created a pipeline by using pipe() function. This pipeline allows two proccess to communicate with each other. In other words, one process' output can be the input of the other process. Then, I called fork() function to create a new child process with its own unique PID which will do the "MAN" operation. I also used wait() function in "SHELL process" to ensure "MAN process" runs before the "SHELL process". I closed the STDOUT_FILENO to prevent the writing to the console. Then, I used dup() function to replace this STDOUT_FILENO with our pipe fd[1]. Then, I used execvp() function to execute "man rm" command and the result of this command has written into the fd[1] pipe because of the replacement. After that, I called fork() function again to create a new child process with its own unique PID which will do the "GREP" operation. I also again used wait() function in "SHELL process" to ensure "GREP" process runs before the "SHELL process". I closed the STDIN_FILENO and again replaced it with our pipe fd[0] by using dup() function. Therefore, the grep command read from the output of MAN process pipe not console. I also closed the STOUT_FILENO to not write to the console but opened "outut.txt" file. Then, I used execvp() function to execute "grep "^ *-v" -w -A 1" command and the result of this command has written into the output.txt. After end of the "GREP" process, the "SHELL" process has printed out its own ID to the console and program terminated.

Process Hierarchy:
SHELL PROCESS (PARENT) -> MAN PROCESS (CHILD) -> SHELL PROCESS (PARENT) -> GREP PROCESS (CHILD) -> SHELL PROCESS(PARENT)

Two child processes used for "MAN" and "GREP" commands and these commands communicated with each other by using pipes.