SHELL

1. First I have created stored the path of Current Working Directory in working_directory, and execution variables.
2. Then I have taken the input from user in terminal, in while loop; which shows them their current directory.
3. After taking the arguments from the user, I have formatted the data as per requirement (eg. use of calloc and malloc, removal of extra spaces if added by mistake, removing “\n” character from command so that the it doesn’t create any issue while executing the process) so that it becomes easy to interpret commands.
4. On the basis of first argument I have made different If-else statements.
5. Internal commands are written inside the shell.c file; while external commands are called by creating a child process and then using execv() command.
6. As per the command given in question, I have changed the current working directory, after the process is executed.
7. We can end the program by writing exit command.

WORD

1. Firstly I have separated secondary commands by If-else statement.
2. For normal command (word input.txt) the code will count all the strings separated by spaces, and return it.
3. I have written code for counting the number of words in such a way that, it uses flag approach, when the pointer, do not points to any word then the flag variable will check, whether the pointer was in the word or not. If it was in the word then the word count increases.
4. For -n command (word -n input.txt) the code will count all the strings separated by spaces and neglect the “\n” character, and return it.
5. For -d command (word -d input1.txt input2.txt) the code will count all the strings separated by spaces in 2 files, and return the difference of words between them.

DIR

1. Firstly, I have created the child process and then called for execv(), command.
2. I have separated secondary commands by If-else statement.
3. For normal command (dir hello), it will first check by opendir() command whether hello directory exists or not, if not then it will create the directory using mkdir() command. And change the current directory to hello.
4. For -r command (dir -r hello), it will first check by opendir() command whether hello directory exists or not, if not then it will create the directory using mkdir() command. And change the current directory to hello. Else it will remove the directory using the deleteDirectoryRecursively() function. And then create a new empty directory named hello, and change the current directory to hello.
5. For -v command (dir -v hello), it will first check by opendir() command whether hello directory exists or not, if not then it will create the directory using mkdir() command. And change the current directory to hello. Else it will change the current directory to hello. It will inform the user at every step, about the process happening.
6. Assumption: After every process the current directory is changed to the new directory created or mentioned by the user

DATE

1. Firstly, I have created the child process and then called for execv(), command.
2. I have separated secondary commands by If-else statement.
3. For Accessing the meta data of file we use stat.h header in C file.
4. For normal command (date input.txt) it will return the Date and Time of last modification of the file, which will be in default format.
5. For -R command (date -R input.txt) it will return the Date and Time of last modification of the file, which will be in format of RFC5322 which will include the timezone.
6. For -d command (date -d input.txt) it will return the Date and Time of last modification of the file, which will be in format of DDth MM YYYY, HH:MM:SS.