Problem: UNIX has no recycle bin at the command line. When you remove a file or directory, it is gone and cannot be restored. The purpose of this project is to write a remove script and a restore script to provide users with a recycle bin which can be used to safely delete and restore files.

Phase 1 - Basic Functionality

Write a script called remove that mimics the rm command. The remove script should be able to accept the name of a file as a command line argument as rm does, but instead of deleting the file your script should move it to a recycle bin directory called 'deleted' in your home directory.

1. The script name is 'remove' and will be stored in $HOME/project.
2. The recycle bin will be $HOME/deleted. If the deleted directory does not already exist, your script must create it.
3. The file to be deleted will be a command line argument and the script should be executed as follows: **bash remove fileName** or **sh remove fileName**
4. The script must test for the following error conditions and display the same error messages as the rm command
    a. File does not exist - Display an error message if file supplied does not exist
    b. No filename provided - Display an error message if no filename is provided as an argument
    c. Directory name provided - Display an error message if a directory name is provided instead of a filename
5. The filenames in the recycle bin, will be in the following format: fileName_inode. For example, if a file named f1 with inode 1234 was removed, the file in the recycle bin will be named f1_1234. This gets around the potential problem of deleting 2 files with the same name. The recycle bin will only contain files, not directories with files.
6. Create a hidden file called .restore.info in $HOME. Each line of this file will contain the name of the file in the recycle bin, followed by a colon, followed by the original full path of the file. For example, if a file called f1, with an inode of 1234, was removed from the /home/trainee1 directory, this file will contain:

    f1_1234:/home/trainee1/f1

    If another file named f1, with an inode of 5432, was removed from the /home/trainee1/testing directory, then restore.info will contain:

    f1_1234:/home/trainee1/f1
    f1_5432:/home/trainee1/testing/f1
7. Test that the file being deleted is not the remove script. If it is, display the error message "Attempting to delete remove - operation aborted" and terminate the script with an exit status. (Be sure to copy remove before testing that this works.)

Phase 2-Basic Restore

Write a script called restore to restore individual files back to their original location. The user will determine which file is to be restored and use the file name with inode number in order to restore the file. For example: **sh restore f1_1234**

1. Script name is 'restore' and stored in $HOME/project
2. The file to be restored will be a command line argument and the script should be executed as follows: **bash restore f1_1234** or **sh restore f1_1234** (This is the name of the file in $HOME/deleted.)
3. The file must be restored to its original location, using the pathname stored in the .restore.info file
4. The script should test for the following error conditions and display similar error messages to the rm command
    a. File does not exist - Display an error message if file supplied does not exist
    b. No filename provided - Display an error message if no file provided as argument
5. The script must check whether the file being restored already exists in the target directory. If it does the user will be asked, "Do you want to overwrite? y/n" The script must restore the file if the user types y, Y, or yes to this prompt and not restore it if they type anything else.
6. After the file has been successfully restored, the entry in the .restore.info file will be removed.

Phase 3-Multiple Files, Wildcards and Option Flags

The rm command can remove multiple files, for example **rm file 1 file2 file3**, and use wildcards, for example **rm f*** . The rm command can use the -i flag, for interactive, and -v flag, for verbose. Add this functionality to your remove

1. Ensure the script can delete multiple files and wildcards.
2. Update the script to test for the command line argument of -i and if used, display a message asking for confirmation, in the same way as rm -i
3. Update the script to test for the command line argument -v and if used, display a message confirming deletion, in the same way as rm -v
4. Ensure the script works with both options in either order, -iv or -vi