Answers to UNIX exercise;

1. Use a text editor (jedit/nedit/gedit/komodo/textwrangler) to create a file mycommands.txt where you write all commands and observations you do in the following exercises. Use copy/paste to copy the commands.
   Note: There are more standard text editors than nedit, etc. Examples are emacs, xemacs, vi, vim, and pico.
   > jedit mycommands.txt

2. First list the files in the directory. You need to be in home directory;
   > ls

3. Copy ex1.acc to myfile.acc. You need to have downloaded the file;
   > cp ex1.acc myfile.acc

4. Look at the content of both files to ensure they are identical.
   > ls -l
   > less ex1.acc
   > less myfile.acc

5. Copy ex1.dat to myfile.acc.
   > cp ex1.dat myfile.acc

6. Check that the content of myfile.acc changed.
   > less myfile.acc

7. Delete myfile.acc
   > rm myfile.acc

8. Make a directory test and move the three files to it.
   > mkdir test
   > mv ex1.acc test
   > mv ex1.dat test
   > mv orphans.sp test

9. Make a directory data and move the three files to that instead.
   Assuming you're in the home directory and not the test directory:
   > mkdir data
   > mv test/ex1.acc data
   > mv test/ex1.dat data
   > mv test/orphans.sp data

10. Remove test directory.
    > rmdir test

11. Change directory to data and confirm that you succeed. Go back to the home directory or work directory afterwards.
> cd data
> ls
> cd ..

12. Make three new directories newtest - one inside the other, like a russian doll.
> mkdir newtest
> mkdir newtest/newtest
> mkdir newtest/newtest/newtest

13. Move the data directory to the innermost newtest directory.
> mv data newtest/newtest/newtest

14. Confirm that the three files are moved along with the data directory
> ls newtest/newtest/newtest/data

15. Copy the three files to your home (your top directory).
> cp newtest/newtest/newtest/data/* .

16. Remove all newtest directories and data in the with a single command.
> rm -rf newtest

17. Count the lines in ex1.acc and ex1.dat.
> wc ex1.acc
> wc ex1.dat

18. Concatenate ex1.acc and ex1.dat in the file ex1.tot, i.e. copy the content of two files into one new file.
Verify that all gene IDs comes first followed by numerical data.
> cat ex1.acc ex1.dat > ex1.tot
> less ex1.tot

19. Merge/Paste ex1.acc and ex1.dat together in ex1.tot, thus destroying the old file.
Verify that corresponding gene IDs and numerical data are put on the same line. as the data.
> paste ex1.acc ex1.dat > ex1.tot
> less ex1.tot

20. Extract (cut) SwissProt ID and 3nd numerical data (column 1 and 5) from ex1.tot. Put results into a file ex1.res.
> cut -f1,5 ex1.tot > ex1.res

21. Find the 3 SwissProt ID's in ex1.res which have the largest number(s) in column 2, i.e. the top 3 entries.
> cut -f2,3 ex1.tot | sort -k2 -r > ex1.res

22. Find the lines (using grep) in orphans.sp which contain a GenBank accession number.
There are 85, verify this. Note: An accession number is one or two capital letters and looks like this 'AB000114.CDS.1', the .CDS. part is kind of optional.
In this and the following exercise you should use a precise pattern. By limiting the "degrees of freedom" on the pattern you find only the relevant lines. An example is very clear in ex. 23; If you just grep for HUMAN in order to find the human swissprot genes then you are MUCH more likely to pick up a wrong line (a line where the word HUMAN is part of the ordinary text) than if you were grepping for _HUMAN. _HUMAN is simply a stronger pattern than HUMAN in this case. Usually a pattern is stronger the longer it is. However, what really matters is the uniqueness of the pattern.

```bash
> grep .CDS. orphans.sp
> grep .CDS. orphans.sp | wc
```

or

```bash
> grep '=>' orphans.sp
> grep '=>' orphans.sp | wc
```

23. How many human genes with SwissProt IDs in orphans.sp exist? How many of those are hypothetical? (11)
How many genes belong to the rat, and how many of those are precursors? (9)
Note: A Swissprot ID looks like 'PARG_HUMAN' or 'TF1A_MOUSE', with the gene being before the underscore and the organism after the underscore.

```bash
> grep _HUMAN orphans.sp | wc
> grep _HUMAN orphans.sp | grep HYPOTHETICAL | wc
> grep _RAT orphans.sp | wc
> grep _RAT orphans.sp | grep PRECURSOR | wc
```

24. This little exercise will require that man is used for help on grep. From the file ex1.res find the lines with positive numbers and put then into ex1.pos. The lines with negative number go into ex1.neg.
Some versions of grep/unix requires -e others does not. Read the man page for grep to find out. It has to do with the way grep understands the dash., ie considers it a part of an option.

```bash
> grep -v - ex1.res > ex1.pos
> grep -e - ex1.res > ex1.neg
```

or

```bash
> grep - ex1.res > ex1.neg
```

25. Write a shell script that solves exercise 19-24, with the exercises clearly seperated in both the script and the output.
This should be straight forward (but long), especially since you took notes (exercise 1).
```bash
echo Exercise 19
paste ex1.acc ex1.dat > ex1.tot
head ex1.tot

> cat ex1.tot
```

```bash
echo Exercise 20
cut -f1,5 ex1.tot > ex1.res
head ex1.res
```

```bash
echo Exercise 21
cut -f2,3 ex1.tot | sort -k2 -r > ex1.res
head ex1.res
```
echo Exercise 22
grep .CDS. orphans.sp
echo Number of Genbank Accessions
grep .CDS. orphans.sp | wc

echo Exercise 23
echo Number of human SWISSPROT genes
grep _HUMAN orphans.sp | wc
echo Number of hypothetical human SWISSPROT genes
grep _HUMAN orphans.sp | grep HYPOTHETICAL | wc
echo Number of rat SWISSPROT genes
grep _RAT orphans.sp | wc
echo Number of precursor rat SWISSPROT genes
grep _RAT orphans.sp | grep PRECURSOR | wc

echo Exercise 24
grep -v -ex1.res > ex1.pos
grep -e -ex1.res > ex1.neg
echo Number of positive numbers counted by wc
wc -l ex1.pos
echo Number of negative numbers counted by wc
wc -l ex1.neg

26. Write a shell script (which is simply just a list of unix commands in a file) that puts all the positive numbers in the file ex1.dat into a file ex1.pos2, and all the negative numbers into a file ex1.neg2.
Column position does not matter. The script must clean up after itself, so if any temporary files are used, they must be deleted as the last action. Remember to put the date and a description of the files in the first lines of the resulting output files.
cut -f1 ex1.dat > numbers
cut -f2 ex1.dat >> numbers
cut -f3 ex1.dat >> numbers
date > ex1.pos2
echo The positive numbers from ex1.dat >> ex1.pos2
grep -v -numbers >> ex1.pos2
date > ex1.neg2
echo The negative numbers from ex1.dat >> ex1.neg2
grep -e -numbers >> ex1.neg2rm numbers

Another solution with lots of temporary files, needed when not appending >>.
cut -f1 ex1.dat > numbers1
cut -f2 ex1.dat > numbers2
cut -f3 ex1.dat > numbers3
date > datefile
echo The positive numbers from ex1.dat > textfile
grep -v - numbers1 > pos1
grep -v - numbers2 > pos2
grep -v - numbers3 > pos3
cat datefile textfile pos1 pos2 pos3 > ex1.pos2
echo The negative numbers from ex1.dat > textfile
grep -e - numbers1 > neg1
grep -e - numbers2 > neg2
grep -e - numbers3 > neg3
cat datefile textfile neg1 neg2 neg3 > ex1.neg2
rm numbers1 numbers2 numbers3 pos1 pos2 pos3 neg1 neg2 neg3 datefile textfile

A solution completely without temporary files.

date > ex1.pos2
echo The positive numbers from ex1.dat >> ex1.pos2
cut -f1 ex1.dat | grep -v >> ex1.pos2
cut -f2 ex1.dat | grep -v >> ex1.pos2
cut -f3 ex1.dat | grep -v >> ex1.pos2

date > ex1.neg2
echo The negative numbers from ex1.dat >> ex1.neg2
cut -f1 ex1.dat | grep -e >> ex1.neg2
cut -f2 ex1.dat | grep -e >> ex1.neg2
cut -f3 ex1.dat | grep -e >> ex1.neg2

For some reason people often forget to add the date and head line to the result file. Perhaps they concentrate on separating the numbers, but the date etc. is also part of the exercise.
Read the text at least twice, once before starting and once after completion.

27. Mail your mycommands.txt file to the teacher for comments.