1. (3 points) Assume you started a new process in the background using 
\textit{xterm} & in the \textit{bash}. Name three different ways to terminate this 
process externally, i.e. without exiting the terminal via \textit{logout} etc.

\begin{itemize}
\item \texttt{kill PID}
\item \texttt{kill}
\item \texttt{kill \%1}
\end{itemize}
2. (3 points) How do these expressions differ when used as a regular expression? \([0-9]^*\) and \([0-9][0-9]^*\)

The second one consists of at least one digit while the first one can be the empty string.
3. (4 + 4 points)

(a) Using `sed`, how do you add the tags `<HTML>` at the beginning and `</HTML>` at the end of a file?

(b) How do you delete all lines with `sed` which begin with /* and end with */?

a) `sed -e '1i <HTML>\n' -e '\$0 <\HTML\n'`

b) `sed 's/\*.*\*/\n'`
4. (3 + 3 points) Implement the following commands in *awk*:

(a) `head -n 5`

(b) `sed -n '5,10p'`

```
a) awk ' NR < 6 { print }'

b) awk ' NR > 4 && NR < 11 { print }'
```
5. (8 points) Write an awk program to provide extra spaces at the end of every line (if required) so that the line length is maintained at 127 (assume there is no line longer than 127 characters).

```awk
{l = length "$0"
 line = "$0 "
 for (i = l; i <= 127; i++) line = line " "
 printf line
}
```
Name:

6. (7 points) From a tar archive print only the pathnames of directories. Directory pathnames end with a /, but the tar output may contain a variable number of fields.

```bash
BEGIN { FS="/" }
line = ""
for (i = 1; i <= NF; i++)
    line = line="/" $i
print line
```
7. (10 points) Write a bash script that looks up every .c file in the current directory for the strings printf or fprintf. If found, the script adds the statement #include <stdio.h> at the beginning of the file but only if it does not already have it included.

```bash
printf='gcc -r -E printf *'
fprintf='gcc -r -E fprintf *'
for f in printf fprintf; do
  if grep "# include <stdio.h>" $f; then
delete nothing to do for file $f
  else
    mv $f $f.tmp
cat $f.tmp | sed '1c # include <stdio.h> '
      > $f
    rm $f.tmp
  fi
done
```
8. (8 points) Devise a script that takes a filename as argument and looks in the home directory tree to display the listing of all links that point to the given filename.

```bash
files=`find ~ -type l -print`
for f in $files; do
    if [ "basename "$f"" = "$1" ]; then
delete "$f"

done
```
9. (2 + 2 points)

(a) You have to run a job at night and need to have both the output and error messages in the same file. How will you run the script?

(b) When running the above job, you would like to logout after starting the script. How can you prevent the script from being terminated during logout?

a) `script > messages 2>&1`

b) `nohup script > messages 2>&1`
10. (4 points) A make rule does not always have a dependency, and the target need not be a disk file. Explain with an example of a makefile entry.

This can be used, for example, as a part of a clean keep method:

```make
all:
  make

clean:
  rm *.o
```