Unix Tools
Courant Institute of Mathematical Sciences
Homework assignment 2 – Solution
March 1, 2007

A dictionary file, `/usr/dict/websters`, is available on your system. Many of the questions below make use of this dictionary.

Except when you are asked to write a script, for each of the following questions, the answer should be given as a single sequence of piped Unix commands. The answers should be given in a single ascii file, which is what you will turn in.

0. create a symbolic link from your working directory with the name `dic` to `/usr/dict/websters`.

```
i5$ ln -s /usr/dict/websters dic
```

1. Find all the words that contain exactly one vowel.

```
i5$ grep -i '^[^aeiou]*[aeiou][^aeiou]*$' dic
A
a
Ab
abb
...
zythum
Zyzomys
```

2. Find all the *squares* in the dictionary, that is words w that can be written as $w = uu$ for some sequence of characters u.

```
i5$ grep '^\(.*\)\1$' dic
aa
adad
akeake
...
wawa
yaya
zoozoo
```

3. Print the list of all bigrams appearing in the dictionary and their number of occurrences, in increasing order of frequency [a bigram is a sequence of two consecutive characters].

```
i5$ nawk 'BEGIN{ FS="" }{for (i=1; i<NF; i++) arr[$i$(i+1)]++} \
END{ for (i in arr) print i, arr[i]}' < dic | sort -k2n
...
en 24916
ic 26442
at 27163
an 27693
al 27816
te 29176
on 29729
ti 31542
in 33567
er 42321
```

4. Define the accumulated count of a bigram as the sum of the counts of all bigrams up to that bigram in the sorted list printed out in the previous exercise, and the frequency of a bigram as the ratio of its accumulated count and the total count of all bigrams. Use the previous question to print out the *median bigrams*, that is the two bigrams with frequencies closest to .5, as well as the accumulated counts and frequencies of these bigrams (the result might seem surprising but I did not cook the answer).

```
i5$ nawk '{ sum+=$2; arr[NR]=sum; str[NR]=$1 } \
END{ for(i=1; i<NR; i++) if((x1=arr[i]/sum) < .5 && \
(x2=arr[i+1]/sum) >= .5) \
print str[i], str[i+1], arr[i], \
x1, arr[i+1], x2 }' < bigram
me di 1000929 0.496261 1012931 0.502212
```

5. By what percentage is the number of distinct words of the dictionary decreased if we remove the following suffixes: *able*, *ing*, *ly*, *tion*, *al*, *ize*, *ness* from the words?

```
i5$ nawk 'BEGIN{ RS="able\n|ing\n|ly\n|tion\n|al\n|ize\n|ness\n"; }\
```

```
{ print $0 }' < dic | \
sort | uniq -c | \
nawk '{ sum+=$1 } END{ print 1 - NR/sum }'
0.0810004
```

6. The utility `spell` (which can be found under `/usr/bin/`) can be used to print plausible derivations of a word, as in:

   ```
   i5$ echo hyperantiaccommodativeness | spell -v
   +hyper+anti-e+ion-ion+ive+ness  hyperantiaccommodativeness
   ```

   We are interested in the morphological sequences preceded with a `+`, e.g. `hyper, anti, ion, ive, ness` in the example above. Print all these morphological sequences in the order of decreasing occurrences in the dictionary as well as the corresponding number of occurrences (*hint*: it could be useful to use the functions `gsub` and `split` under `nawk`).

   ```
   i5$ spell -v dic | \
   nawk 'NF>1{ str=$1; gsub(/-[^+]*/,"", str);\
   n=split(str, a, "+"); for(i=2; i<=n; i++) print a[i] }' | \
   sort | uniq -c | sort -k1nr
   11895 un
   7712 ly
   5649 ing
   5084 ness
   4324 ed
   3839 er
   3319 ist
   3273 able
   3238 al
   3157 ion
   3129 d
   3094 re
   2523 non
   2232 ize
   2162 pre
   2071 less
   2053 r
   2020 over
   ```

```
1496 ive
1289 dis
1247 like
...
```

7. Write a bash script called `myspell` that takes as argument two file names `filename1` and `filename2`. `filename1` must be the name of an existing ascii file. `filename2` is generated from `filename1` by surrounding words that were not found in the dictionary by < and >. The script should be as safe as possible, in particular it should check that the desired arguments are provided.

```
#!/bin/bash

USAGE="Usage:
    $0 [-?] filename1 filenam2
Options:
   -?           help"

while getopts \? c
do
        case $c in
        \?) echo "$USAGE" ; exit 0 ;;
        esac
done

shift `expr $OPTIND - 1`

if [ $# -ne 2 ]
then
   echo "$USAGE"
   exit 0
fi

file1=$1
file2=$2

if !([ -f $file1 ]) || !([ -r $file1 ])
then
echo "$0 error: $file1 is not a regular file or is not readable."
```

```
exit 1
fi

gawk '{ for (i=1; i<=NF; i++)
if(system("egrep -i '\''^"$i"$'\'' dic > /dev/null"))
printf("<%s> ", $i)
else
printf("%s ", $i);
printf("\n") }' $file1
```