Software Construction

args.sh

A simple shell script demonstrating access to arguments.

```
echo My name is $0
echo My process number is $$
echo I have $# arguments
echo My arguments separately are $*
echo My arguments together are "$@"
echo My 5th argument is "'$5'"
```

l

l [file|directories...] - list files

Short Shell scripts can be used for convenience.

Note: "$@" like $* expands to the arguments to the script, but preserves the integrity of each argument if it contains spaces.

```
ls -las "$@"
```

word_frequency.sh

Count the number of time each different word occurs in the files given as arguments, e.g. word_frequency.sh dracula.txt

```
sed 's/ /\n/g' "$@"|       # convert to one word per line
tr A-Z a-z|                # map uppercase to lower case
sed "s/[^a-z']//g"|        # remove all characters except a-z and '
egrep -v '^$'|             # remove empty lines
sort|                      # place words in alphabetical order
uniq -c|                   # use uniq to count how many times each word occurs
sort -n                    # order words in frequency of occurrance
```

iota.v1.sh

Print the integers 1..n if 1 argument given.

Print the integers n..m if 2 arguments given.

```
if test $# = 1
then
    start=1
    finish=$1
elif test $# = 2
then
    start=$1
    finish=$2
else
    echo "Usage: $0 <start> <finish>" 1>&2
    exit 1
fi

for argument in "$@"
do
    # clumsy way to check if argument is a valid integer
    if echo "$argument"|egrep -v '^-?[0-9]+$' >/dev/null
    then
        echo "$0: argument '$argument' is not an integer" 1>&2
        exit 1
    fi
done

number=$start
while test $number -le $finish
do
    echo $number
    number=`expr $number + 1`     # or number=$(($number + 1))
done
```

iota.v2.sh

Print the integers 1..n if 1 argument given.
Print the integers n..m if 2 arguments given.

Using bash arithmetic which is more reabable but less portable

```
if ((# == 1))
then
    start=1
    finish=$1
elif ((# == 2))
then
    start=$1
    finish=$2
else
    echo "Usage: 0 <start> <finish>" 1>&2
    exit 1
fi

for argument in "@"
do
    # This use of a regex is a bash extension missing from many Shells
    # It should be avoided if portability is a concern
    if ! [[ "argument" =~ ^-?[0-9]+ ]]
    then
        echo "0: argument 'argument' is not an integer" 1>&2
        exit 1
    fi
done

number=start
while ((number <= finish))
do
    echo number
    number=$((number + 1))
done
```

tolower.sh

Change the names of the specified files to lower case.
Note the use of test to check if the new filename differs from the old.
The perl utility rename provides a more general alternative.

Note without the double quotes below filenames containing spaces would be handled incorrectly.

Note also the use of -- to avoid mv interpreting a filename beginning with - as an option

Although a files named -n or -e will break the script because echo will treat them as an option,

```
if test # = 0
then
    echo "Usage 0: <files>" 1>&2
    exit 1
fi

for filename in "@"
do
    new_filename=`echo "filename" | tr A-Z a-z`
    test "filename" = "new_filename" && continue
    if test -r "new_filename"
    then
        echo "0: new_filename exists" 1>&2
    elif test -e "filename"
    then
        mv -- "filename" "new_filename"
    else
        echo "0: filename not found" 1>&2
    fi
done
```

watch_website.sh

Repeatedly download a specified web page until a specified regexp matches its source then notify the specified email address.

For example:

```
repeat_seconds=300  #check every 5 minutes

if test $# = 3
then
    url=$1
    regexp=$2
    email_address=$3
else
    echo "Usage: $0 <url> <regex>" 1>&2
    exit 1
fi

while true
do
    if wget -O- -q "$url"|egrep "$regexp" >/dev/null
    then
        echo "Generated by $0" | mail -s "$url now matches $regexp" $email_address
        exit 0
    fi
    sleep $repeat_seconds
done
```

create_1001_file_C_program.sh
create 1001 C files, compile and runs them
file f$i.c contains a defintion of function f$i which returns $i for example file42.c will contain a function f42 that returns 42
main.c contains code to call all 1000 functions and print the sum of their return values

add the initial lines to main.c note the use of quotes on eof to disable variable interpolation in the here document

```
cat >main.c <<'eof'
#include <stdio.h>

int main(void) {
    int v = 0 ;
eof

i=0
while test $i -lt 1000
do
    # add a line to main.c to call the function f$i

    cat >>main.c <<eof
    int f$i(void);
    v += f$i();
eof

    # create file$i.c containing function f$i

    cat >file$i.c <<eof
int f$i(void) {
    return $i;
}
eof

    i=$((i + 1))
done

cat >>main.c <<'eof'
    printf("%d\n", v);
    return 0;
}
eof

# compile and run the 1001 C files

time clang main.c file*.c
./a.out
```

plagiarism_detection.simple_diff.sh

Run as plagiarism_detection.simple_diff.sh <files>

Report if any of the files are copies of each other

The use of diff -iw means changes in white-space or case won't affect comparisons

```
for file1 in "$@"
do
    for file2 in "$@"
    do
        test "$file1" = "$file2" && break
        if diff -i -w "$file1" "$file2" >/dev/null
        then
            echo "$file1 is a copy of $file2"
        fi
    done
done
```

plagiarism_detection.comments.sh

Improved version of plagiarism_detection.simple_diff.sh

The substitution s/\/\/.*// removes // style C comments.
This means changes in comments won't affect comparisons.

Note use of temporary files

```
TMP_FILE1=/tmp/plagiarism_tmp1$$
TMP_FILE2=/tmp/plagiarism_tmp2$$


for file1 in "$@"
do
    for file2 in "$@"
    do
        if test "$file1" = "$file2"
        then
            break # avoid comparing pairs of assignments twice
        fi
        sed 's/\/\/.*//' "$file1" >$TMP_FILE1
        sed 's/\/\/.*//' "$file2" >$TMP_FILE2
        if diff -i -w $TMP_FILE1 $TMP_FILE2 >/dev/null
        then
            echo "$file1 is a copy of $file2"
        fi
    done
done
rm -f $TMP_FILE1 $TMP_FILE2
```

plagiarism_detection.identifiers.sh

Improved version of plagiarism_detection.comments.sh

This version converts C strings to the letter 's' and it converts identifiers to the letter 'v'.
Hence changes in strings & identifiers won't prevent detection of plagiarism.

The substitution s/"["]*"/s/g changes strings to the letter 's'
This pattern won't match a few C strings which is fine for our purposes

The s/[a-zA-Z_][a-zA-Z0-9_]*/v/g changes all variable names to 'v' which means changes to variable names won't affect comparison.
Note this also may change function names, keywords etc.

This is fine for our purposes.

```
TMP_FILE1=/tmp/plagiarism_tmp1$$
TMP_FILE2=/tmp/plagiarism_tmp2$$
substitutions='s/\/\/.*//;s/"[^"]"/s/g;s/[a-zA-Z_][a-zA-Z0-9_]*/v/g'

for file1 in "$@"
do
    for file2 in "$@"
    do
        test "$file1" = "$file2" && break # don't compare pairs of assignments twice
        sed "$substitutions" "$file1" >$TMP_FILE1
        sed "$substitutions" "$file2" >$TMP_FILE2
        if diff -i -w $TMP_FILE1 $TMP_FILE2 >/dev/null
        then
            echo "$file1 is a copy of $file2"
        fi
    done
done
rm -f $TMP_FILE1 $TMP_FILE2
```

plagiarism_detection.reordering.sh

Improved version of plagiarism_detection.identifiers.sh

Note the use of sort so line reordering won't prevent detection of plagiarism.

```
TMP_FILE1=/tmp/plagiarism_tmp1$$
TMP_FILE2=/tmp/plagiarism_tmp2$$
substitutions='s/\/\/.*//;s/"[^"]"/s/g;s/[a-zA-Z_][a-zA-Z0-9_]*/v/g'

for file1 in "$@"
do
    for file2 in "$@"
    do
        test "$file1" = "$file2" && break # don't compare pairs of assignments twice
        sed "$substitutions" "$file1"|sort >$TMP_FILE1
        sed "$substitutions" "$file2"|sort >$TMP_FILE2
        if diff -i -w $TMP_FILE1 $TMP_FILE2 >/dev/null
        then
            echo "$file1 is a copy of $file2"
        fi
    done
done
rm -f $TMP_FILE1 $TMP_FILE2
```

plagiarism_detection.md5_hash.sh

Improved version of plagiarism_detection.reordering.sh

Note use md5sum to calculate a Cryptographic hash of the modified file http://en.wikipedia.org/wiki/MD5 and then use sort && uniq to find files with the same hash

This allows execution time linear in the number of files

```
substitutions='s/\/\/.*//;s/"[^"]"/s/g;s/[a-zA-Z_][a-zA-Z0-9_]*/v/g'

for file in "$@"
do
    echo `sed "$substitutions" "$file"|sort|md5sum` $file
done|
sort|
uniq -w32 -d --all-repeated=separate|
cut -c36-
```

local.sh

print print numbers < 10000 demonstrate use of local Shell builtin to scope a variable
without the local declaration below the variable i in the function would be global and would break the bottom while loop
local is not (yet) POSIX but is widely supported

```
is_prime() {
    local n i
    n=$1
    i=2
    while test $i -lt $n
    do
        test $((n % i)) -eq 0 && return 1
        i=$((i + 1))
    done
    return 0
}

i=0
while test $i -lt 1000
do
    is_prime $i && echo $i
    i=$((i + 1))
done
```

repeat_message.sh
demonstrate simple use ofa shell function

```
repeat_message() {
    n=$1
    message=$2
    for i in $(seq 1 $n)
    do
        echo "$i: $message"
    done
}

i=0
while test $i -lt 4
do
    repeat_message 3 "hello Andrew"
    i=$((i + 1))
done
```

where.v0.sh

Printall occurances of executable programs with the specified names in $PATH
Note use of tr to produce a space-separated list of directories suitable for a for loop.
Breaks if directories contain spaces (fixing this left as an exercise).

```
if test $# = 0
then
    echo "Usage $0: <program>" 1>&2
    exit 1
fi

for program in "$@"
do
    program_found=''
    for directory in `echo "$PATH" | tr ':' ' '`
    do
        f="$directory/$program"
        if test -x "$f"
        then
            ls -ld "$f"
            program_found=1
        fi
    done
    if test -z $program_found
    then
        echo "$program not found"
    fi
done
```

where.v1.sh

Print all occurances of executable programs with the specified names in $PATH
Note use of tr to produce a list of directories one per line suitable for a while loop.
Won't work if directories contain spaces (fixing this left as an exercise)

```
if test $# = 0
then
    echo "Usage $0: <program>" 1>&2
    exit 1
fi

for program in "$@"
do
    echo "$PATH"|
    tr ':' '\n'|
    while read directory
    do
        f="$directory/$program"
        if test -x "$f"
        then
            ls -ld "$f"
        fi
    done|
    egrep '.' || echo "$program not found"
done
```

where.v2.sh

Print all occurances of executable programs with the specified names in $PATH
Note use of tr to produce a list of directories one per line suitable for a while loop.
Won't work if directories contain new-lines (fixing this left as an exercise)

```
if test $# = 0
then
    echo "Usage $0: <program>" 1>&2
    exit 1
fi
for program in "$@"
do
    n_path_components=`echo $PATH|tr -d -c :|wc -c`
    index=1
    while test $index -le $n_path_components
    do
        directory=`echo "$PATH"|cut -d: -f$index`
        f="$directory/$program"
        if test -x "$f"
        then
            ls -ld "$f"
            program_found=1
        fi
        index=`expr $index + 1`
    done
    test -n $program_found || echo "$program not found"
done
```

COMP(2041|9044) 20T2: Software Construction is brought to you by
the School of Computer Science and Engineering
at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs2041@cse.unsw.edu.au
CRICOS Provider 00098G