CITS2003/CITS4407 End of Semester Test Questions (1½ hrs; Total 60 marks)

Q1. The Awk script `gday.awk` contains the following **[2 marks]**:

```
BEGIN{
  array[0] = "Hullo"
  array[1] = "Hullo"
}

{for(i=0; i<=2; i++)
  print array[i] " " $1
}
```

When the script is invoked with the command:
`echo Fred | awk -f gday.awk` which of the following happens:

a) You see "Hullo Fred" twice
b) You see "Hullo Fred" three times
c) You see "Hullo Fred" twice, followed by "Fred"
d) There is a runtime error because a variable has been referenced without having previously been assigned a value.

Ans: C (1 mark if A is chosen)

Q2. You have been given an Awk script to fix. The script aims to sum all the numbers in a file, i.e. both across the lines of the file and also down the file. Each line can have multiple values and can vary in length. Unfortunately, the script is not working as intended.

```
{
 for(i=1; i< NF; i++)
  sum = $i
}
END{
  printf("sum = %d\n",sum)
}
```

When you run the script on a file of integers,

```
1 2 3
4 5 6
7 8 9 10
```

a) The sum was 9 (rather than 55). What caused that problem? **[4 Marks]**

Ans: sum = $i should be sum += $i (or sum = sum + $i)

b) When that error has been fixed, you rerun the code and are informed that the sum is 36. Still not right. What is the source of this problem? **[4 Marks]**

Ans: The loop test i < NF should be i <= NF (It is okay if they report the bugs in the wrong order)

Q3.

a) Write a Sed command that takes a file name containing spaces, and replaces each space with an underscore "_". **[2 Marks]**

Ans: s/ /_/g

b) Write a Sed command that adds the string '`.faa`' to a file name **[2 Marks]**

Ans: s/$/.faa/

c) Write a Sed command that adds "../data/" to the start of each file name. **[3 marks]**

Ans: s/^/..\/data\//

Q4. Write a runnable Bash script, `do_mvs` <directory> which, given two directories, looks for ordinary files (i.e. not directories) in the first directory (and recursively through its subdirectories) , and moves each of the files to the second directory. For example, assume directory `level1` contains file `a` and directory `level2`, and directory `level2` contains file `b`:

level1
    a
    level2
        b

then after `do_mvs(level1, other_directory)` the files `a` and `b` have been moved, `other_directory` now has:

a
b

while `level1` now looks like:

level 1
    level 2

The program will first need to check that the files given as the argument exist and are directories. (Hint. You may wish to consider using the `find` command.) **[12 marks]**

Ans:

```
!/usr/bin/env bash
if [[ $# -eq 0 || ! -d $1  || ! -d $2 ]]
then
```

```
  echo "usage: $0 <directory> <directory>" > /dev/stderr
  exit 0
fi

find $1 -type f -exec mv '{}' $2 \;
```

Mark allocation

- #!/ etc 2 marks
- If statement(s) with 3 tests 4 marks
- Sensible error message and exit 2
- Sensible call to find 4 (don't sweat the punctuation)

Q5.

a. Does the string `rotor` match regular Ed-style regular expression `r..r` **[2 Marks**

It Matches It Does Not Match

b. Does the string `abababababc` match regular Ed-style regular expression `^[ab]*$` **[2 Marks]**

It Matches It Does Not Match

c. Does the string `abababababc` match regular Ed-style regular expression `[ab]*c[ab]*` **[2 Marks]**

It Matches It Does Not Match

Q6. Write one or more Ed style regular expressions that completely match all of the following lines. Use as few regular expressions as possible. Please note that the regular expression must involve the letters 'a', 'b', 'c', 'd' and 'e', but not '.'

```
a b c d
a a c d
a c d
a c e
```

a) What is/are the regular expression(s) **[4 Marks]**

Ans: `aa*b*c[de]`

b) In words, explain how do your regular expressions can be used to match all four lines. **[4 Marks]**

Ans: The first a is found in all lines, so is in the RE. a can then appear at least once more, hence a*. b does follow in one line, but mostly not, so b*. (Actually b? but that was not taught.) Then compulsory c then either d or e.

Q7. Sabine is using git to track her work. She has just updated a script called configure.sh. The old file looked like this:

configure.sh

```
#!/bin/bash

if [[ $1 -gt 9000 ]]
then
a="stardust"
else
a="duchess"
fi

device_code=$(grep "$a" codes.csv | cut -d, -f2)
./activate.sh $device_code $1
```

The updated file looks like this:

configure.sh

```
#!/bin/bash

if [[ "$1" =~ ^[1-9][0-9]*$ ]]
then
    power=$1
else
    echo "Usage: ./configure.sh <power> # power must be a
positive integer"
    exit 1
fi

if [[ $power -gt 9000 ]]
then
    device="stardust"
else
    device="duchess"
fi

device_code=$(grep "$device" codes.csv | cut -d, -f2)
./activate.sh $device_code $power
```

a) What command (or commands) should Sabine use to record her changes in git? **[2 Marks]**

Ans: git add configure.sh (1 mark)
git commit (1 mark)
OR
git commit -a (2 marks)
OR
git add -A (1 mark)
git commit (1 mark)

b). Write a suitable git commit comment to describe Sabine's changes. **[3 Marks]**

Ans: something along the line of input validation/antibugging, improved code readability,

c). Git is a useful tool for group projects and individual projects. What is one advantage of git when used for an individual project? **[2 Marks]**

Ans: Any of
- git helps you understand past code changes
- git helps you know which version of your code is most recent
- git allows you to back up your code remotely (no need to mention github but no penalty for doing so)
- git allows you to easily revert mistakes/restore a working version

Q8. A computer science department noticed that there an unusual number of submissions for assignments on particular days. Luckily the department's assignment submission system kept a log of submissions per day. You have been asked to write code to report the student IDs and the number of submissions for those unusual submissions on a particular day. The data you've been given is in a <tab> separated file with fields resembling:
<*date*> <TAB> <*time*> <TAB> <*ID*> <TAB> <*Assessment*>

You can ignore the first column as we know the data comes from a single day. Also, to keep things simple the time is just recorded as an integer, so 7:02 is 702, while 23:00 (11pm) is recorded as 2300.
For example,

```
today     900  james     Ass2
today     2302 james     Ass2
```

The input data is sorted by increasing time of the day. With that in mind, your code should look for 3 or more submissions for the assessment item **Ass2** occurring after 11pm (**23:00**), and report the corresponding IDs and number of submissions. The list of IDs should be in alphabetical order. (Don't worry about adding antibugging code.)

Hint: You can structure this as a single Awk script, `very_suby.awk`, or an Awk script with a short Bash script `very_suby.sh` **[10 Marks]**

Ans

```
# Expected format: Date Time ID Assessment
$2 >= 2300 {if($4 == "Ass2")
          subs_id[$3]++
}

END{
  for(id in subs_id)
    if(subs_id[id] >= 4)
```

```
        printf("%s\t%s\n", id, subs_id[id]) | "sort -k 1n"
}
```

Another (more likely) solution is to create an Awk script, but without the internal call sort, and place the call to the awk script in a Bash script, with the call to Awk followed by a call to sort.

Marks breakdown:

- 4 marks for gathering subs which are post 23:00 and for Ass2
- 3 for looping through to get those with at least 4 hits
- 3 for the final sort