H.Dip Data Analytics
- GMIT -

Computer Architecture and Technology Convergence Assignment

Gareth Duffy - (G00364693)

Table of contents

Q1: Binary arithmetic

Q1.1

Binary addition:

	1		1	1	← Carry
		1	0	1	1
+	1	1	0	1	1
= 1	0	0	1	1	0
** 32	16	8	4	2	1

Decimal addition:

	1	1
+	2	7
	3	8

**** Bit weights in bottom row for decimal conversion**

Thus, 1011 + 11011 = 100110 in binary (38 in decimal).

Q1.2

Two's compliment of -31:

1.	**31 in binary:**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**
2.	**Invert all bits:**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**0**
3.	**Two's compliment integer:**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**

Add one

Two's compliment of -59:

1.	**59 in binary:**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**1**
2.	**Invert all bits:**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**
3.	**Two's compliment integer:**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**1**

Add one

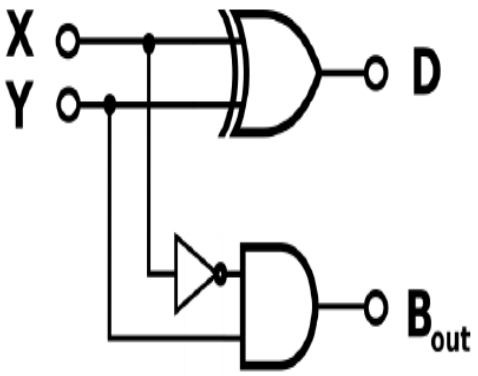
Q1.3

The bit pattern 11101001 represents 00010111 if we interpret it as an 8-bit two's compliment integer, and 23 if we convert it to decimal:

Bit pattern:	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
Invert all bits:	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**
Two's compliment integer:	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**
****Bit weight:**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Confirm bit Weight:				**Yes**		**Yes**	**Yes**	**Yes**

By adding the bit weights, we get 23 in decimal: **16 + 4 + 2 + 1 = 23.

Q1.4



Logic diagram for half subtractor

Input	Input	Output	Output
X	**Y**	**Sum D**	**Carry B**
0	**0**	**0**	**0**
0	**1**	**1**	**1**
1	**0**	**1**	**0**
1	**1**	**0**	**0**

Truth table for half subtractor

This circuit diagram is known as a half subtractor. Regarding its function, it is a combinational circuit used to perform subtraction of two bits. We can see it is comprised of two inputs and two outputs. Two inputs correspond to two typical input bits and two outputs corresponds to the *difference* bit (D) and the *borrow-out* bit (B).

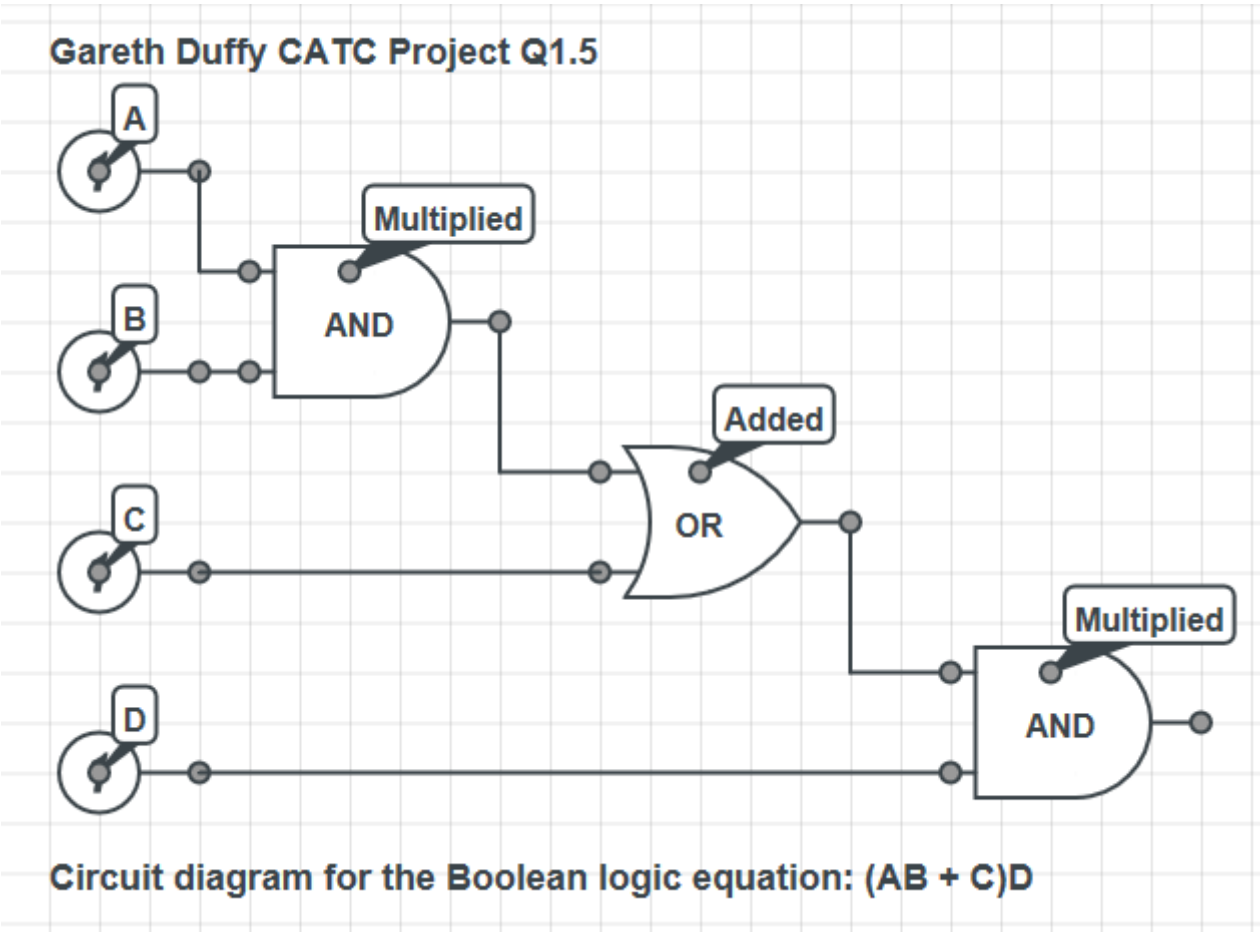
The binary subtraction is performed by the Ex-OR gate with additional circuitry to perform the borrow operation. Thus, the half subtractor is designed via an Ex-OR gate, including an AND gate with the X input *complimented* (to represent its invert/negative) before being fed to the AND gate.

From the truth table we can see that the difference (D) output is the result of the Ex-OR gate and the borrow-out (Bout) is the result of the NOT-AND combination.

Essentially, it is very similar to a half adder with only one difference in input X, i.e. the *minuend* which is complimented by the NOT gate before being applied at the AND gate to implement the borrow/carry output (B). The Y input is known as the *subtrahend*, i.e. the binary digit which is to be subtracted [1].

Q1.5

Circuit diagram of Boolean logic equation [3]:



Q2: Linux assignment

Q2.1

Description of what each Linux command does:

“echo hello world”

Echo displays a line of text with whatever you provide it, e.g. it will echo the strings to standard output (similar to the “print” command in Python).

“passwd”

This command changes passwords for user accounts. A *normal user* may only change the password for his/her own account, while the *superuser* may change the password for any account. “passwd” also changes the account or associated password validity period.

“date”

This command shows the current date and time in the given format or sets the system date.

“hostname”

Used to display or set the system’s DNS name (Domain Name System). DNS serves as the phone book for the Internet by translating human-friendly computer hostnames into IP addresses. This command also displays or set its hostname or NIS name (Network Information Service). NIS a client–server directory service protocol for distributing system configuration data such as user and host names between computers on a computer network.

“arch”

This will display the machine architecture name.

“uname -a”

Displays certain system information. In the case of -a, it will print all system information in a particular order.

“dmesg | more”

Used to print, examine or control the kernel ring buffer. The kernel ring buffer is a data structure that records messages related to the operation of the kernel. A ring buffer is a special kind of buffer that is always a constant size, removing the oldest messages when new messages come in.

“uptime”

Gives a one-line display of the following information:

The current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

“whoami”

Displays the username associated with the current effective user ID, i.e. who you are logged in as.

“who”

Displays information about users who are currently logged in.

“last”

Shows a listing of users who were last logged in.

Essentially, “last” searches back through the /var/log/wtmp file (or the file designated by the -f option) and displays a list of all users logged in (and out) since that file was created. One or more usernames and/or ttys can be given, in which case last will show only the entries matching those arguments. Names of ttys can be abbreviated, thus last 0 is the same as last tty0.

(A tty command in Linux is one that can be entered interactively or as part of a script to determine whether the output for the script is a terminal (that is, to an interactive user) or to some other destination such as another program or a printer. “tty” meant *TeleTYpewriter* originally and now also means any terminal or serial port on Linux/Unix systems.

“finger”

This is a user information lookup program which displays information about the system users, e.g. the user's login name, real name, terminal name and write status.

“w”

Shows who is logged on and what they are doing. i.e. displays information about the users currently on the machine, and their processes.

“top”

The top program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel. The types of system summary information shown and the types, order and size of information displayed for processes are all user configurable and that configuration can be made persistent across restarts.

“echo $SHELL”

Displays the name of the environment variable that holds your current preferred (default) shell (in this case /bin/bash, i.e. the Borne Again Shell), not the current running shell which would be called with the “echo $0” command.

“echo {con, pre}{sent, fer}{s, ed}”

Takes the input and combines the syllables in the brackets to create words.

“man ls”

Displays the user manual information pertaining to listing directory contents, i.e. information about the files (the current directory by default). It also sorts entries alphabetically if no format is specified.

“man who”

Displays the user manual information pertaining to printing information about users who are currently logged in.

“**clear**”

Clears the terminal screen.

“cal 2000”

Displays a simple calendar, in this case that of the year 2000. If no arguments are specified, the current month will be displayed.

“cal 9 1752”

Displays the ninth month (September) of the year 1752. However, something unusual about this month stood out immediately; Eleven days are missing from the calender (3rd till the 13th).

After subsequent research I discovered that this is because at the time, England shifted from the Roman Julian calender to the Gregorian calender under The British Calender Act of 1751, which proclaimed that in British (and American colonies) Thursday 3rd September 1752 should become Thursday the 14th. The king of England thus ordered that these 11 days be wiped from the month of September 1752. The employees worked for 11 days less, but still got paid for the entire 30 days. This is how the term “Paid leave” was born, and historically speaking, nothing whatsoever occurred in British history between the 3rd and 13th of September 1752.

“yes please”

“yes” outputs an affirmative response, or a user-defined string (in this case “please”) of text continuously until killed. For example, typed by itself, the yes command outputs 'y' or whatever is specified as an argument, followed by a newline repeatedly until stopped by the user or otherwise killed.

“time sleep 5”

The command “time” shows you how long a command will take to run. Here, we used the “sleep” command which ran from approximately 5 seconds (real 0m5.001s), as shown in the output screenshot below.

“history”

Displays an indexed output of the command history with the oldest command listed first (See the output screenshot below).

```
garethduffy@ip-172-31-28-234: ~
*** System restart required ***
Last login: Thu Apr 26 19:51:13 2018 from 37.228.242.107
garethduffy@ip-172-31-28-234:~$ time sleep 5

real    0m5.001s
user    0m0.000s
sys     0m0.000s
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ history
  697  # Redirecting command output to a new file.
  698  echo `w` >> garethduffy.txt
  699  # history: Shows you all of the last commands that have been recently used.
  700  history
  701  # Redirecting command output to a new file.
  702  echo `history` >> garethduffy.txt
  703  echo | <enter>
  704  # top: Displays Linux processes
  705  top
  706  cat garethduffy.txt
  707  /
  708  tree
  709  ls garethduffy.txt
  710  cat
  711  cat garethduffy.txt
  712  ls-l
  713  ls-| q
  714  ls-l
  715  ls-l garethduffy
  716  cd..
  717  /
  718  tree
  719  ls -l /garethduffy]
  720  ls -l /garethduffy
  721  ls-l garethduffy
  722  ls-l
  723  ls -l
  724  chmod 700 garethduffy
  725  chmod
  726  chmod 700
  727  chmod help
  728  'chmod --help'
  729  help chmod
  730  man chmod
  731  chmod 700 /garethduffy
  732  ls -l
  733  chmod 700 garethduffy
```

Screenshot of “history” command output.

Many programs read input from the user a line at a time. The GNU History library is able to keep track of those lines, associate arbitrary data with each line, and utilize information from previous lines in composing new ones. By default, the history command will show you the last five hundred commands you have entered.

Q2.2

Shell script output redirected to txt file:

Gareth Duffy (G00364693) CATC, GMIT

Assignment Q2.2; Shell script to automate execution of commands.

This script will output the following set of information and redirect each one to text file: garethduffy.txt

Today's date is: Thu Apr 26 19:53:37 UTC 2018

The current IP address is: ip-172-31-28-234

The current machine architecture is: x86_64

System information is Linux ip-172-31-28-234 4.4.0-1049-aws #58-Ubuntu SMP Fri Jan 12 23:17:09 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

This system has been running for: 19:53:37 up 45 days, 20:23, 6 users, load average: 0.00, 0.00, 0.00

The username associated with this account is:garethduffy

The following users are logged in: garethduffy pts/0 2018-04-26 19:51 (37.228.242.107)

davidobrien pts/1 2018-04-26 19:34 (185.51.72.60)

marcomen pts/2 2018-04-26 17:33 (86.45.17.193)

aideenbyrne pts/3 2018-04-26 17:48 (89.127.16.14)

patrickmeehan pts/4 2018-04-26 17:55 (86.40.75.36)

marcomen pts/5 2018-04-26 18:29 (86.45.17.193)

Below is information pertaining to current users: Login Name Tty Idle Login Time Office Office Phone

aideenbyrne pts/3 1:50 Apr 26 17:48 (89.127.16.14)

davidobrien pts/1 18 Apr 26 19:34 (185.51.72.60)

garethduffy pts/0 Apr 26 19:51 (37.228.242.107)

marcomen pts/2 2:10 Apr 26 17:33 (86.45.17.193)

marcomen pts/5 1 Apr 26 18:29 (86.45.17.193)

patrickmeehan pts/4 3 Apr 26 17:55 (86.40.75.36)

The following users are logged in this is their activity 19:53:37 up 45 days, 20:23, 06 users, load average: 0.00, 0.00, 0.00

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

garethdu pts/0 37.228.242.107 19:51 0.00s 0.08s 0.00s w

davidobr pts/1 185.51.72.60 19:34 18:47 0.03s 0.03s -bash

marcomen pts/2 86.45.17.193 17:33 2:10m 0.13s 0.10s vi tea.sh

aideenby pts/3 89.127.16.14 17:48 1:50m 0.05s 0.05s -bash

patrickm pts/4 86.40.75.36 17:55 3:20 0.12s 0.12s -bash

marcomen pts/5 86.45.17.193 18:29 1:21 0.06s 0.06s -bash

The following list commands have been recently used:

690 echo `who` >> garethduffy.txt

691 # finger: Displays information about the system users.

692 echo `finger`

693 # Redirecting command output to a new file.

694 echo `finger` >> garethduffy.txt

695 # w: Shows who is logged on and what they are doing.

696 echo `w`

697 # Redirecting command output to a new file.

698 echo `w` >> garethduffy.txt

699 # history: Shows you all of the last commands that have been recently used.

700 history

701 # Redirecting command output to a new file.

```
702  echo `history` >> garethduffy.txt
703  echo | <enter>
704  # top: Displays Linux processes
705  top
706  cat garethduffy.txt
707  /
708  tree
709  ls garethduffy.txt
710  cat
711  cat garethduffy.txt
712  ls-l
713  ls-| q
714  ls-l
715  ls-l garethduffy
716  cd..
717  /
718  tree
719  ls -l /garethduffy]
720  ls -l /garethduffy
721  ls-l garethduffy
722  ls-l
723  ls -l
724  chmod
700 garethduffy
725  chmod
726  chmod
700
727  chmod help
728  'chmod --help'
729  help chmod
730  man chmod
731  chmod 700 /garethduffy
```

```
732  ls -l
733  chmod 700 garethduffy
734  chmod 700 garethduffy.txt
735  ls -l
736  chmod 700 Music
737  ls - l
738  ls-l
739  ls -l
740  lynx www.google.com
741  # Gareth Duffy (G00364693) GMIT CATC
742  # The Mrs. Doyle While Loop
743  # Shell script that behaves like an Irish person offering a cup of tay
744  input = ""
745  echo "Will you have a cup of tea? (yes / no)"
746  read input
747  yes
748  # Gareth Duffy (G00364693) GMIT CATC
749  # The Mrs. Doyle While Loop
750  # Shell script that behaves like an Irish person offering a cup of tay
751  inp = ""
752  echo "Will you have a cup of tea? (yes / no)"
753  read inp
754  # The Mrs. Doyle While Loop
755  # Shell script that behaves like an Irish person offering a cup of tea
756  inp = " "
757  echo "Will you have a cup of tea? (Type: yes or no)"
758  read inp
759  yes
760  # Gareth Duffy (G00364693) GMIT CATC
761  # The Mrs. Doyle While Loop
762  # Shell script that behaves like an Irish person offering a cup of tea
763  inp = " "
```

```
764  echo "Will you have a cup of tea? (Type: yes or no)"

765  read inp

766  ad inp

767  if [$inp = "yes"]; then ow"

768  elif [$inp = "no"]; then

769  p

770  if [$inp = "yes"]; then

771  ow"; elif [$inp = "no"]; then  p; if [$inp = "yes"]; then ow"

772  elif [$inp = "no"]; then

773  p

774  if [$inp = "yes"]; then

775  ow"; elif [$inp = "no"]; then  e; n; q; exit; help;

776  while true; do     read -p "Do you wish to install this program?" yn;     case $yn in        [Yy]* ) make install; break;;        [Nn]* ) exit;;        * ) echo "Please answer yes or no.";;     esac; done

777  while true; do     read -p "Will ye have a cup of tea?" yn;     case $yn in        [Yy]* ) make install; break;;        [Nn]* ) exit;;        * ) echo "Please answer yes or no.";;     esac; done

778  while true; do     read -p "Will ye have a cup of tea?" yn;     case $yn in        [Yy]* ) make install; break;;        [Nn]* ) echo "Are you sure";;        * ) echo "Please answer yes or no.";;     esac; done

779  while true; do     read -p "Will ye have a cup of tea?" yn;  if [$yn = "yes"]; then ow"

780  elif [$yn = "no"]; then

781  while true; do     read -p "Will ye have a cup of tea?" [yn] answer;  if [$answer = "yes"]; then ow"

782  elif [$answer = "no"]; then

783  echo "Do you like pie?"

784  read pie

785  n

786  read -p "Will you have a cup of tea ? (yes/no)" reply

787  yes

788  #!/bin/bash

789  Whilw tre; do

790  read -p "Will you have a cup of tea ? (yes/no)" reply

791  while true; do     read -p "Will ye have a cup of tea?" yn;     case $yn in        [Yy]* ) echo "Great, I'll make tea now";;        [Nn]* ) echo "Are you sure";;        * ) echo "Please answer yes or no.";;     esac; done

792  While true; do

793  read -p "Are you alright? (y/n) " RESP
```

794 While true; do

795 yes

796 While true; do

797 read -p "Will ye have a cup of tea?" yn

798 no

799 While true; do

800 read -p "Will ye have a cup of tea?" yn

801 echo "Will you have a cup of tea? (Type: yes or no)" inp

802 read inp

803 inp = ""

804 echo "Will you have a cup of tea? (Type: yes or no)" inp

805 read inp

806 While true; do

807 read -p "Will you have a cup of tea? (Type: yes or no)" inp

808 no

809 While true; do

810 read -p "Will you have a cup of tea? (Type: yes or no)" yn

811 echo "Do that? [Y,n]"

812 read input

813 y

814 echo "Do that? [Y,n]"

815 read input

816 echo "Do that? [Y,n]"

817 read input

818 if [[$input == "Y" || $input == "y"]]; then echo "do that"; else echo "don't do that"; fi

819 echo "Do that? [Y,n]"

820 read input

821 if [[$input == "Y" || $input == "y"]]; then echo "do that"; else echo "don't do that"; fiy; y; bash; echo "Do that? [Y,n]"; read input; if [[$input == "Y" || $input == "y"]]; then echo "do that"; else echo "don't do that"; fi;

822 echo "Do that? [Y,n]"

823 read input

824 echo "Do that? [Y,n]"

825 read input

826 echo "Does a wall needs to be sent?"

827 read input

828 no"]; then

829 exit

830 fi

831 input=""

832 echo "Does a wall needs to be sent?"

833 read input

834 while true; do read -p 'Continue? (y/n): ' answer; case "$answer" in [Yy]*) printf "%s\n" 'Looping once more.';; [Nn]*) printf "%s\n" 'Bailing out!'; exit 0;; *) printf "%s\n" 'Answer either “y” or “n”.'; esac; done

835 while true; do read -p 'Continue? (y/n): ' answer; case "$answer" in [Yy]*) printf "%s\n" 'Looping once more.';; [Nn]*) printf "%s\n" 'Bailing out!'; exit 0;; *) printf "%s\n" 'Answer either “y” or “n”.'; esac; done

836 while true; do read -p "Will ye have a cup of tea? (y/n)" answer; case "$answer" in [Yy]*) printf "%s\n" "Great, I'll make tea now";; [Nn]*) printf "%s\n" "Are you sure?"; *)

837 done

838 while true; do read -p "Will ye have a cup of tea? (y/n)" answer; case "$answer" in [Yy]*) printf "%s\n" "Great, I'll make tea now";; [Nn]*) printf "%s\n" "Are you sure?"; exit 0;; *) printf "%s\n" "Answer either “y” or “n”."; esac; done

839 while true; do read -p "Will ye have a cup of tea? (y/n)" answer; case "$answer" in [Yy]*) printf "%s\n" "Great, I'll make tea now";; [Nn]*) printf "%s\n" "Are you sure?"; printf "%s\n" "Answer either “y” or “n”.";; *) printf "%s\n" "Answer either “y” or “n”."; esac; done

840 # Correct version

841 While true; do

842 # Correct version

843 While true; do

844 while true; do read -p 'Continue? (y/n): ' answer; case "$answer" in [Yy]*) printf "%s\n" 'Looping once more.';; [Nn]*) printf "%s\n" 'Bailing out!'; exit 0;; *) printf "%s\n" 'Answer either “y” or “n”.'; esac; done

845 function ask { echo $1 # add this line; read -n 1 -r; if [[$REPLY =~ ^[Yy]$]]; then return 1; else exit; echo "Abort.."; fi; }

846 function ask { echo $1 # add this line; read -n 1 -r; if [[$REPLY =~ ^[Yy]$]]; then return 1; else exit; echo "Abort.."; fi; While true; do

847 While true; do

848 down vote

849 I noticed that no one posted an answer showing multi-line echo menu for such simple user input so here is my go at it:

850 #!/bin/bash

851 function ask_user() { echo -e "

852 #~~~~~~~~~~~~#

853 | 1.) Yes |

854 | 2.) No |

855 | 3.) Quit |

856 #~~~~~~~~~~~~#\n"; read -e -p "Select 1: " choice; if ["$choice" == "1"]; then do_something; elif ["$choice" == "2"]; then do_something_else; elif ["$choice" == "3"]; then clear && exit 0; else echo "Please select 1, 2, or 3." && sleep 3; clear && ask_user; fi; }

857 ask_user

858 2

859 3

860 #!/bin/bash

861 # This script will test if you have given a leap year or not.

862 echo "Will you have a cup of tea?" (y/n)

863 read (y/n)

864 if (((y/n) == "yes")); then echo "great, Ill make tea"; else echo "Are you sure?"; fi

865 read -p "Continue (y/n)?" choice

866 #!/bin/sh

867 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

868 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; fi

869 #!/bin/sh

870 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

871 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; fi

872 #!/bin/sh

873 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

874 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; if [Nn]; then echo " Ah go on"; fi; garethduffy;

875 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

876 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; if [Nn]; then echo " Ah go on"; fi; q

877 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

878 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; elif; then

879 fi

880 # Correct version

881 While true; do

882 done

883 #!/bin/sh

884 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) "Great, I'll make tea now";; [Nn]*) "Are you sure?";; *) echo "Please answer yes or no.";; esac; done; }

885 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; fi

886 #!/bin/sh

887 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

888 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; fi

889 #!/bin/sh

890 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

891 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; then

892 fi

893 #!/bin/sh

894 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

895 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure";

896 #!/bin/sh

897 promptyn () { while true; do read -p "$1 " yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) echo "Please answer yes or no.";; esac; done; }

898 if promptyn "Will ye have a cup of tea?"; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; then

```
- 899  fi
900  #!/bin/sh
901  promptyn () {     while true; do        read -p "$1 " yn;        case $yn in           [Yy]* ) return 0;;           [Nn]*
) return 1;;           * ) echo "Will ye have a cup of tea?";;        esac;     done; }
902  if promptyn [Yy]; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
903  y
904  #!/bin/sh
905  promptyn () {     while true; do        read -p "$1 " yn;        case $yn in           [Yy]* ) return 0;;           [Nn]*
) return 1;;           * ) echo "Will ye have a cup of tea?";;        esac;     done; }
906  if promptyn; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
907  #!/bin/sh
908  promptyn () {     while true; do        read -p "$1 " yn;        case $yn in           [Yy]* ) return 0;;           [Nn]*
) return 1;;           * ) echo "Will ye have a cup of tea?";;        esac;     done; }
909  if promptyn; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
910  #!/bin/sh
911  promptyn () {     while true; do        read -p "Will ye have a cup of tea?" yn;        case $yn in           [Yy]* )
return 0;;           [Nn]* ) return 1;;           * ) echo "Will ye have a cup of tea?";;        esac;     done; }
912  if promptyn; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
913  #!/bin/sh
914  promptyn () {     while true; do        read -p "Will ye have a cup of tea?" yn;        case $yn in           [Yy]* )
return 0;;           [Nn]* ) return 1;;           * ) echo "Will ye have a cup of tea?";;        esac;     done; }
915  if promptyn; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
916  #!/bin/sh
917  promptyn () {     while true; do        read -p "Will ye have a cup of tea?" yn;        case $yn in           [Yy]* )
return 0;;           [Nn]* ) return 1;;           * ) ;;        esac;     done; }
918  if promptyn; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
919  #!/bin/sh
920  promptyn () {     while true; do        read -p "Will ye have a cup of tea?" yn;        case $yn in           [Yy]* )
return 0;;           [Nn]* ) return 1;;           * ) ;;        esac;     done; }
921  if promptyn; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
922  history
923  #!/bin/sh
924  promptyn () {     while true; do        read -p "Will ye have a cup of tea?" yn;        case $yn in           [Yy]* )
return ;;           [Nn]* ) return ;;           * ) ;;        esac;     done; }
925  if promptyn; then     echo "Great, I'll make a cup of tea"; else     echo "Are you sure"; fi
```

```
#!/bin/sh
promptyn () {    while true; do       read -p "Will ye have a cup of tea?" yn;       case $yn in          [Yy]* ) return ;;          [Nn]* ) return ;;          * ) ;;       esac;    done; }
if promptyn; then    echo "Great, I'll make a cup of tea"; else    echo "Are you sure"; fi
#!/bin/sh
promptyn () {    while true; do       read -p "Will ye have a cup of tea?" yn;       case $yn in          [Yy]* ) return ;;          [Nn]* ) return ;;          * ) ;;       esac;    done; }
if promptyn; then    echo "Great, I'll make a cup of tea"; else    echo "Are you sure"; fi
#!/bin/sh
promptyn () {    while true; do       read -p "Will ye have a cup of tea?" yn;       case $yn in          [Yy]* ) return 0;;          [Nn]* ) return 1;;          * ) ;;       esac;    done; }
if promptyn; then    echo "Great, I'll make a cup of tea"; else    echo "Are you sure"; if promptyn; then echo "Great, I'll make a cup of tea"; else    echo "Are you sure"; fi;
#!/bin/sh
promptyn () {    while true; do       read -p "Will ye have a cup of tea?" yn;       case $yn in          [Yy]* ) return 0;;          [Nn]* ) return 1;;          * ) ;;       esac;    done; }
if promptyn; then    echo "Great, I'll make a cup of tea"; else    echo "Are you sure";    if [Yy]; then echo "Great, I'll make a cup of tea";    else       echo "Ah go on"; fi;
#!/bin/sh
promptyn () {    while true; do       read -p "Will ye have a cup of tea?" yn;       case $yn in          [Yy]* ) return 0;;          [Nn]* ) return 1;;          * ) ;;       esac;    done; }
if promptyn; then    echo "Great, I'll make a cup of tea"; else    echo "Are you sure";    if promptyn; then echo "Great, I'll make a cup of tea";    else       echo "Ah go on"; fi;
#!/bin/sh
promptyn () {    while true; do       read -p "Will ye have a cup of tea?" yn;       case $yn in          [Yy]* ) return 0;;          [Nn]* ) return 1;;          * ) ;;       esac;    done; }
if promptyn; then    echo "Great, I'll make a cup of tea"; else    echo "Are you sure"; if promptyn; then echo "Great, I'll make a cup of tea"; else    echo "Ah go on"; fi;
while [[ x$COMPANY != xyes && x$COMPANY != xno ]]; do echo Please answer yes or no; read COMPANY; done
#!/usr/bin/env bash
# While running an infinite loop
while [ 1 ]; do    echo "Is that all sir? (Yes/No): "    read word    if [ $word = "Yes" ]; then       break;    fi; done
#!/usr/bin/env bash
# While running an infinite loop
```

950 while [1]; do echo "Will you have a cup of tea? (Yes/No): " read word if [$word = "Yes"]; then break; else echo "Are you sure?"; fi; done

951 #!/usr/bin/env bash

952 # While running an infinite loop

953 while [1]; do echo "Will you have a cup of tea? (Yes/No): " read word if [$word = "Yes"]; then break; else echo "Are you sure?"; fi; done

954 #!/usr/bin/env bash

955 # While running an infinite loop

956 while [1]; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

957 #!/usr/bin/env bash

958 # While running an infinite loop

959 while [1]; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

960 #!/usr/bin/env bash

961 # While running an infinite loop

962 while [1]; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then break; else echo "Are you sure?"; fi; done

963 #!/usr/bin/env bash

964 # While running an infinite loop

965 while [1]; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; if [$word = "yes"]; then echo "Ah go on"; fi; done

966 #!/usr/bin/env bash

967 # While running an infinite loop

968 while [1]; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; if [$word = "yes"]; then echo "Ah go on"; fi; done

969 #!/usr/bin/env bash

970 # While running an infinite loop

971 while [1]; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

972 #!/usr/bin/env bash

973 # While running an infinite loop

974 while true; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

975 #!/usr/bin/env bash

976 # While running an infinite loop

977 while true; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

978 #!/usr/bin/env bash

979 # While running an infinite loop

980 while true; do echo "Will you have a cup of tea?" read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

981 #!/usr/bin/env bash

982 # While running an infinite loop

983 while true; do echo "Will you have a cup of tea?" read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

984 #!/usr/bin/env bash

985 # While running an infinite loop

986 while true; do echo "Will you have a cup of tea?" read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

987 #!/usr/bin/env bash

988 # While running an infinite loop

989 while true; do echo "Will you have a cup of tea?" read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

990 #!/usr/bin/env bash

991 # While running an infinite loop

992 while true; do read -p "Will you have a cup of tea?" read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

993 #!/usr/bin/env bash

994 # While running an infinite loop

995 while [1]; do echo "Will you have a cup of tea? (yes/no): " read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

996 #!/usr/bin/env bash

997 # While running an infinite loop

998 while; do

999 t, I'll make tea"

1000 break

1001 else

1002 echo "Are you sure?"

1003 fi

1004 done

1005 #!/usr/bin/env bash

1006 # While running an infinite loop

1007 while true; do echo "Will you have a cup of tea?"; break read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

1008 #!/usr/bin/env bash

1009 # While running an infinite loop

1010 while true; do read -p "Will you have a cup of tea?" read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

1011 #!/usr/bin/env bash

1012 # While running an infinite loop

1013 while true; do echo "Will you have a cup of tea?"; read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

1014 #!/bin/bash

1015 while true; do echo "Will you have a cup of tea?" ; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi;

1016 #!/bin/bash

1017 while true; do echo "Will you have a cup of tea?" ; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1018 #!/bin/bash

1019 while true; echo "Will you have a cup of tea?" ; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1020 #!/bin/bash

1021 while true; do echo "Will you have a cup of tea?" ; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1022 #!/bin/bash

1023 while true; do echo "Will you have a cup of tea?" ; break; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1024 yes

1025 #!/bin/bash

1026 while true; do echo "Will you have a cup of tea?" ; fi

1027 #!/bin/bash

1028 while true; do echo "Will you have a cup of tea?" ; fi

1029 while read word; do echo "Will you have a cup of tea?"; if [$word = "yes"] ; then echo "Great, I'll make tea"; elif [$word "no"]; then echo "Are you sure"; fi; done

1030 while read word; do echo "Will you have a cup of tea?"; if [$word = "yes"] ; then echo "Great, I'll make tea"; elif [$word = "no"]; then echo "Are you sure"; fi; done

1031 while ["$month" != "q"]; do echo "Please enter a month (q to quit)"; read month; if ["$month" = "q"]; then echo "Now leaving loop"; elif ["$month" = "January"]; then ...; else echo “This month does not exist.”; fi; done

1032 #!/bin/bash

1033 while true; do echo "Will you have a cup of tea?"; : ; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1034 MAIN() { read -p "Will you have a cup of tea" HEY; while true; do if ["$HEY" == "yes"]; then echo "Great, I'll make tea"; return; else echo "Are you sure?"; MAIN; fi; done; }

1035 MAIN

1036 #!/bin/bash

1037 while true; do echo "Will you have a cup of tea?"; return; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1038 #!/bin/bash

1039 while true; do echo "Will you have a cup of tea?"; fi

1040 done

1041 #!/bin/bash

1042 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1043 #!/bin/bash

1044 while; do

1045 done

1046 #!/bin/bash

1047 while [0]; do echo "Will you have a cup of tea?"; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1048 #!/bin/bash

1049 while true; do echo "Will you have a cup of tea?"; read word; done

1050 #!/bin/bash

1051 while true; echo "Will you have a cup of tea?"; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi; done

1052 #!/bin/bash

1053 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; fi;

1054 #!/bin/bash

1055 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; if [$word = "yes"]; then echo "ah go on"; fi; done

1056 #!/bin/bash

1057 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; if [$word = "yes"]; then echo "ah go on"; fi; done

1058 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "yes"]; then echo "Great, I'll make tea"; break; elif [$word = "no"]; then echo "Are you sure?"; if [$word = "yes"]; then echo "ah go on"; fi;

1059 #!/bin/bash

1060 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; if [$word = "n"]; then echo "ah go on"; if [$word = "y"]; then echo "Great, I'll make tea"; break;

1061 #!/bin/bash

1062 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; if [$word = "n"]; then echo "ah go on"; if [$word = "y"]; then echo "Great, I'll make tea"; break; fi; done

1063 #!/bin/bash

1064 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; if [$word = "n"]; then echo "ah go on"; if [$word = "y"]; then echo "Great, I'll make tea"; break; fi; done

1065 #!/bin/bash

1066 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; if [$word = "n"]; then echo "ah go on"; elif [$word = "y"]; then echo "Great, I'll make tea"; break; fi; done

1067 #!/bin/bash

1068 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; if [$word = "n"]; then echo "ah go on"; elif [$word = "y"]; then echo "Great, I'll make tea"; break; fi; done

1069 #!/bin/bash

1070 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; if [$word = "n"]; then echo "ah go on"; elif [$word = "y"]; then echo "Great, I'll make tea"; break; fi; done

1071 #!/bin/bash

1072 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; fi; done

1073 #!/usr/bin/env bash

1074 # While running an infinite loop

1075 while true; do echo "Will you have a cup of tea?" read word if [$word = "yes"]; then echo "Great, I'll make tea"; break; else echo "Are you sure?"; fi; done

1076 #!/bin/bash

1077 while true; do echo "Will you have a cup of tea?"; read word; done

1078 ---

1079 #!/bin/sh

1080 promptyn () { while true; do read -p "Will ye have a cup of tea?" yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) ;; esac; done; }

1081 if promptyn; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; fi

1082 #!/bin/sh

1083 while true; do read -p "Will ye have a cup of tea?" yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) ;; esac; if [$yn = "y"]; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; fi; done

1084 while true; do read -p "Will ye have a cup of tea?" yn; case $yn in [Yy]*) 0;; [Nn]*) 1;; *) ;; esac; if [$yn = "y"]; then echo "Great, I'll make a cup of tea"; else echo "Are you sure"; fi; done

1085 #!/bin/sh

1086 while true; do read -p "Will ye have a cup of tea?" yn; case $yn in [Yy]*) return 0;; [Nn]*) return 1;; *) ;; esac; if [$yn = "y"]; then echo "Great, I'll make a cup of tea"; break; else echo "Are you sure"; fi; done

1087 #!/bin/bash

1088 while true; do echo "Will you have a cup of tea?"; read word; if [$word = "y"]; then echo "Great, I'll make tea"; break; elif [$word = "n"]; then echo "Are you sure?"; fi; done

1089 #!/usr/bin/env bash

1090 # While running an infinite loop

1091 while false; do echo "Will you have a cup of tea? " read word if [$word = "y"]; then echo "Great, I'll make tea" break; elif [$word = "n"]; then echo "Are you sure?"; fi; done

1092 # While running an infinite loop

1093 while false; do read -p "Will you have a cup of tea?" read word if [$word = "y"]; then echo "Great, I'll make tea" break; elif [$word = "n"]; then echo "Are you sure?"; fi; done

1094 # While running an infinite loop

1095 while true; do echo "Will you have a cup of tea?" read word if [$word = "y"]; then echo "Great, I'll make tea" break; elif [$word = "n"]; then echo "Are you sure?"; fi; done

1096 # While running an infinite loop

1097 while [1] ; do echo "Will you have a cup of tea?" read word if [$word = "y"]; then echo "Great, I'll make tea" break; elif [$word = "n"]; then echo "Are you sure?"; fi; done

1098 # While running an infinite loop

1099 while echo "Will you have a cup of tea?" read word if [$word = "y"]; then echo "Great, I'll make tea" break; elif [$word = "n"]; then echo "Are you sure?"; fi; done

1100 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; *) echo "Please answer yes or no.";; esac; done

1101 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; if [$yn = n]

1102 done

1103 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1104 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; if [$yn = y];then

1105 done

1106 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; if [$yn = y];then

1107 done

1108 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; if [$yn = y];then

1109 done

1110 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; if [$yn = [Nn]];then

1111 done

1112 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1113 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) read -p "Are you sure?" yn; esac; done

1114 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) read -p "Are you sure?" yn; esac; done

1115 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) read -p "Are you sure?" yn; if [$yn = "n"];then echo "Ah go on" ; esac

1116 done

1117 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) read -p "Are you sure?" yn; esac; done

1118 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) read -p "Are you sure?" yn; break;; read -p "Ah go on" yn

1119 done

1120 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1121 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1122 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; {1..4};; esac; done

1123 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; {1..4};; esac; done

1124 repeat = "Are you sure"

1125 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1126 repeat = "Are you sure" {1..3}

1127 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1128 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; repeat 3 do while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;;; done

1129 done

1130 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; repeat 3 {while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;;};;

1131 done

1132 for i in {1..4}; while true; do

1133 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; until [1..4];;

1134 done

1135 n=0

1136 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1137 Nn=0

1138 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1139 Nn=0

1140 Nn=0

1141 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1142 for try in {1..3} ; do [[-d Nn]] && break; done

1143 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1144 for try in {1..3} ; do [[yn]] && break; done

1145 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1146 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; for try in {1..3} ; do break;;

1147 done

1148 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; for try in {1..3} ; do break; esac

1149 done

1150 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1151 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; for try in {1..3} ; do

1152 done

1153 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; for try in {1..3} ; do

1154 done

1155 for try in {1..3} ; do [["Are you sure"]] && break; while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done;

1156 for try in {1..3}; do [[Yn]] && break; while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done;

1157 for try in {1..3} ; do [[Nn]] && break; done

1158 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1159 for try in {1..3} ; do [[yn]] && break; done

1160 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1161 for try in {1..3} ; do [[$yn]] && break; done

1162 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1163 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; for try in {1..3} do break;;

1164 done

1165 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; for try in {1..3} do; break;;

1166 done

1167 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1168 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; if {1..3} do break;;

1169 done

1170 if [[Nn] = {1..3}]; then break; while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done;

1171 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; if [[Nn] = {1..3}]; then

1172 done

1173 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; if [[Nn] = {1..3}]; then break ; esac

1174 done

1175 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1176 echo "Will you have a cup of tea?"

1177 read word

1178 while true; do read -p "Will you have a cup of tea?" yn; read word; if [[$word = y]]; then echo "Great, I'll make tea"; else echo "Are you sure?"; fi;

1179 while true; do read -p "Will you have a cup of tea?" ; read word; if [$word = y]; then echo "Great, I'll make tea"; else echo "Are you sure?"; fi;

1180 while true; do read -p "Will you have a cup of tea?" ; read word; if [$word = y]; then echo "Great, I'll make tea"; else echo "Are you sure?"; fi; done

1181 while true; do read -p "Will you have a cup of tea?" ; read word; if [$word = y]; then echo "Great, I'll make tea"; elif [$word = n]; then echo "Are you sure?"; fi; done

1182 while true; do read -p "Will you have a cup of tea?" ; read word; if [$word = y]; then echo "Great, I'll make tea"; elif [$word = n]; then echo "Are you sure?"; fi; done

1183 while true; do read -p "Will you have a cup of tea?" yn; read word; if [[$word = y]]; then echo "Great, I'll make tea"; break; elif [[$word = n]]; then echo "Are you sure?"; fi; done

1184 while true; do read -p "Will you have a cup of tea?" yn; read yn; if [[$yn = y]]; then echo "Great, I'll make tea"; break; elif [[$yn = n]]; then echo "Are you sure?"; fi; done

1185 while true; do echo "Will you have a cup of tea?" ; fi

1186 while true; do echo "Will you have a cup of tea?" yn; read yn; if [[$yn = y]]; then echo "Great, I'll make tea"; break; elif [[$yn = n]]; then echo "Are you sure?"; fi; done

1187 while true; do echo "Will you have a cup of tea?" ; read yn; if [[$yn = y]]; then echo "Great, I'll make tea"; break; elif [[$yn = n]]; then echo "Are you sure?"; done

1188 while true; do echo "Will you have a cup of tea?" ; read yn; if [[$yn = y]]; then echo "Great, I'll make tea"; break; elif [[$yn = n]]; then echo "Are you sure?"; done

1189 while true; do echo "Will you have a cup of tea?" ; read yn; if [[$yn = y]]; then echo "Great, I'll make tea"; break; elif [[$yn = n]]; then echo "Are you sure?"; fi; done

1190 while true; do echo "Will you have a cup of tea?" ; read yn; if [[$yn = y]]; then echo "Great, I'll make tea"; break; elif [[$yn = n]]; then echo "Are you sure?"; fi; done

1191 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; {1..3};; esac; done

1192 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; [Nn] {1..3};; esac; done

1193 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; for [Nn] {1..3};;

1194 done

1195 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; [$yn] {1..3};;

1196 done

1197 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; [$Nn] {1..3};;

1198 done

1199 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?"; [$yn] {1..3};; esac; done

1200 while true; do read -p "Will you have a cup of tea?" yn; case $yn in [Yy]*) echo "Great, I'll make tea"; break;; [Nn]*) echo "Are you sure?";; esac; done

1201 echo $SHELL

1202 echo {con, pre}{sent, fer}{s, ed}

1203 man ls

1204 man who

1205 clear

1206 man echo

1207 man passwd

1208 man date

1209 man hostname

1210 man arch

1211 man uname -a

1212 man dmesg | more

1213 man uptime

```
1214  man whoami
1215  man who
1216  man last
1217  man finger
1218  man w
1219  man top
1220  man tp
1221  man top
1222  man echo $SHELL
1223  man echo {con, pre}{sent, fer}{s, ed}
1224  echo {con,pre}{sent,fer}{s,ed}
1225  clear
1226  echo $SHELL
1227  echo {con,pre}{sent,fer}{s,ed}
1228  man ls
1229  man man ls
1230  man history
1231  cat garethduffy.txt
1232  q
1233  # Gareth Duffy (G00364693) GMIT
1234  # Assigment Q2.2; Shell script to automate execution of commands.
1235  # This script will return the following set of information output and
1236  # redirect each output to a text file named: garethduffy.txt
1237  # To print the contents of this file in Linux type: cat garethduffy.txt
1238  # date - Prints the current date and time:
1239  echo "Today's date is: `date`"
1240  # Redirecting command output to a new file.
1241  echo `date` > garethduffy.txt
1242  # hostname - Prints hostname information i.e. current IP address:
1243  echo "The current IP address is: `hostname`"
1244  # Redirecting command output to a new file.
1245  echo `hostname` >> garethduffy.txt
```

```
# arch - Prints the machine kernel arcitecture info:
echo "The current machine architecture is: `arch`"
# Redirecting command output to a new file.
echo `arch` >> garethduffy.txt
# uname -a - Prints all system information in a specific order:
echo "System information is `uname -a`"
# Redirecting command output to a new file.
echo `uname -a` >> garethduffy.txt
# uptime - Tells the user how long the system has been running:
echo "This system has been running for: `uptime`"
# Redirecting command output to a new file.
echo `uptime` >> garethduffy.txt
# whoami - Prints  the  username  associated  with the current effective user ID:
echo "The username associated with this account is:`whoami`"
# Redirecting command output to a new file.
echo `whoami` >> garethduffy.txt
# who - Prints information about users who are currently logged in:
echo "The following users are logged in: `who`"
# Redirecting command output to a new file.
echo `who` >> garethduffy.txt
# finger - Displays information about the system users:
echo "Below is information pertaining to current users: `finger`"
# Redirecting command output to a new file.
echo `finger` >> garethduffy.txt
# w - Shows who is logged on and what they are doing:
echo "The following users are logged in this is their activity `w`"
# Redirecting command output to a new file.
echo `w` >> garethduffy.txt
# history - Shows you all of the last commands that have been recently used:
echo "The following list commands have been recently used: `history`"
# Redirecting command output to a new file.
echo `history` >> garethduffy.txt
```

```
1278  echo | <enter>
1279  # top - Displays Linux processes:
1280  #
1281  echo "The following processes are running: `top`"
1282  cat garethduffy.txt
1283  > garethduffy.tx
1284  cat garethduffy.txt
1285  clear
1286  >garethduffy.txt
1287  cd..
1288  cd.
1289  dir
1290  /
1291  dir
1292  tree
1293  rm garethduffy.tx
1294  tree
1295  rm a dynamic real-time view of running system.
1296  tree
1297  rm ip location service
1298  /
1299  cd -
1300  cd-
1301  cd.
1302  cd..
1303  cd:
1304  cd/
1305  cd $HOME
1306  tree
1307  rm ses.
1308  tree
1309  rm ip location service
```

```
1310  tree
1311  cd
1312  dir
1313  rm plays\ informatiom\ about\ the\ users\ currently\ on\ the\ machine,\ and
1314  tree
1315  dir
1316  rm plays
1317  rm running\
1318  tree
1319  rm ip location service
1320  rm iprm
1321  tree
1322  rm ip\ location\ service
1323  tree
1324  rm how\ a\ listing\ of\ last\ logged\ in\ users
1325  tree
1326  a\ dynamic\ real-time\ view\ of\ a\ running\ system.
1327  tree
1328  rm plays\ information\ about\ the\ users\ currently\ on\ the\ machine,\ and\
1329  tree
1330  rm a\ dynamic\ real-time\ view\ of\ a\ running\ system.\
1331  tree
1332  cat garethduffy.txt
1333  cd garethduffy.txt
1334  dir garethduffy.txt
1335  tree
1336  cat Music
1337  dir Music
1338  tree
1339  ls garethduffy.txt
1340  # Gareth Duffy (G00364693) CATC, GMIT
1341  # Assigment Q2.2; Shell script to automate execution of commands.
```

```
# This script will return the following set of information output and
# redirect each output to a text file named: garethduffy.txt
# To print the contents of this file in Linux type: cat garethduffy.txt
# date - Prints the current date and time:
echo "Today's date is: `date`"
# Redirecting command output to a new file.
echo `date` > garethduffy.txt
# hostname - Prints hostname information i.e. current IP address:
echo "The current IP address is: `hostname`"
# Redirecting command output to a new file.
echo `hostname` >> garethduffy.txt
# arch - Prints the machine kernel arcitecture info:
echo "The current machine architecture is: `arch`"
# Redirecting command output to a new file.
echo `arch` >> garethduffy.txt
# uname -a - Prints all system information in a specific order:
echo "System information is `uname -a`"
# Redirecting command output to a new file.
echo `uname -a` >> garethduffy.txt
# uptime - Tells the user how long the system has been running:
echo "This system has been running for: `uptime`"
# Redirecting command output to a new file.
echo `uptime` >> garethduffy.txt
# whoami - Prints  the  username  associated  with the current effective user ID:
echo "The username associated with this account is:`whoami`"
# Redirecting command output to a new file.
echo `whoami` >> garethduffy.txt
# who - Prints information about users who are currently logged in:
echo "The following users are logged in: `who`"
# Redirecting command output to a new file.
echo `who` >> garethduffy.txt
# finger - Displays information about the system users:
```

```
1374  echo "Below is information pertaining to current users: `finger`"
1375  # Redirecting command output to a new file.
1376  echo `finger` >> garethduffy.txt
1377  # w - Shows who is logged on and what they are doing:
1378  echo "The following users are logged in this is their activity `w`"
1379  # Redirecting command output to a new file.
1380  echo `w` >> garethduffy.txt
1381  # history - Shows you all of the last commands that have been recently used:
1382  echo "The following list commands have been recently used: `history`"
1383  # Redirecting command output to a new file.
1384  echo `history` >> garethduffy.txt
1385  echo | <enter>
1386  # top - Displays Linux processes:
1387  echo "The following processes are running: `top`"
1388  q
1389  clear
1390  cat garethduffy.txt
1391  q
1392  tree
1393  ls garethduffy
1394  ls garethduff.txt
1395  ls garethduffy.txt
1396  dir
1397  cat garethduffy.txt
1398  clear
1399  tree
1400  echo -n > garethduffy.txt
1401  cat garethduffy.txt
1402  cat Music
1403  ls garethduffy.txt
1404  cat garethduffy.txt
1405  # Gareth Duffy (G00364693) CATC, GMIT
```

```
# Assigment Q2.2; Shell script to automate execution of commands.
# This script will return the following set of information output and
# redirect each output to a text file named: garethduffy.txt
# To print the contents of this file in Linux type: cat garethduffy.txt
# date - Prints the current date and time:
echo "Today's date is: `date`"
# Redirecting command output to a new file.
echo `date` > garethduffy.txt
# hostname - Prints hostname information i.e. current IP address:
echo "The current IP address is: `hostname`"
# Redirecting command output to a new file.
echo `hostname` >> garethduffy.txt
# arch - Prints the machine kernel arcitecture info:
echo "The current machine architecture is: `arch`"
# Redirecting command output to a new file.
echo `arch` >> garethduffy.txt
# uname -a - Prints all system information in a specific order:
echo "System information is `uname -a`"
# Redirecting command output to a new file.
echo `uname -a` >> garethduffy.txt
# uptime - Tells the user how long the system has been running:
echo "This system has been running for: `uptime`"
# Redirecting command output to a new file.
echo `uptime` >> garethduffy.txt
# whoami - Prints  the  username  associated  with the current effective user ID:
echo "The username associated with this account is:`whoami`"
# Redirecting command output to a new file.
echo `whoami` >> garethduffy.txt
# who - Prints information about users who are currently logged in:
echo "The following users are logged in: `who`"
# Redirecting command output to a new file.
echo `who` >> garethduffy.txt
```

```
1438  # finger - Displays information about the system users:
1439  echo "Below is information pertaining to current users: `finger`"
1440  # Redirecting command output to a new file.
1441  echo `finger` >> garethduffy.txt
1442  # w - Shows who is logged on and what they are doing:
1443  echo "The following users are logged in this is their activity `w`"
1444  # Redirecting command output to a new file.
1445  echo `w` >> garethduffy.txt
1446  # history - Shows you all of the last commands that have been recently used:
1447  echo "The following list commands have been recently used: `history`"
1448  # Redirecting command output to a new file.
1449  echo `history` >> garethduffy.txt
1450  echo | <enter>
1451  # top - Displays Linux processes:
1452  echo "The following processes are running: `top`"
1453  ls garethduffy.txt
1454  cat garethduffy.txt
1455  tree
1456  cat garethduffy.txt
1457  echo -n > garethduffy.txt
1458  cat garethduffy.txt
1459  clear
1460  # date - Prints the current date and time:
1461  # Redirecting command output to a new file.
1462  echo "Today's date is: `date`" > garethduffy.txt
1463  # hostname - Prints hostname information i.e. current IP address:
1464  # Redirecting command output to a new file.
1465  echo "The current IP address is: `hostname`" >> garethduffy.txt
1466  cat garethduffy.txt
1467  echo -n > garethduffy.txt
1468  cat garethduffy.txt
1469  # Gareth Duffy (G00364693) CATC, GMIT
```

```
# Assigment Q2.2; Shell script to automate execution of commands.
# This script will return the following set of information output and
# redirect each output to a text file named: garethduffy.txt
# To print the contents of this file in Linux type: cat garethduffy.txt
# date - Prints the current date and time:
# Redirecting command output to a new file.
echo "Today's date is: `date`" > garethduffy.txt
# hostname - Prints hostname information i.e. current IP address:
# Redirecting command output to a new file.
echo "The current IP address is: `hostname`" >> garethduffy.txt
# arch - Prints the machine kernel arcitecture info:
# Redirecting command output to a new file.
echo "The current machine architecture is: `arch`" >> garethduffy.txt
# uname -a - Prints all system information in a specific order:
# Redirecting command output to a new file.
echo "System information is `uname -a`" >> garethduffy.txt
# uptime - Tells the user how long the system has been running:
# Redirecting command output to a new file.
echo "This system has been running for: `uptime`" >> garethduffy.txt
# whoami - Prints  the  username  associated  with the current effective user ID:
# Redirecting command output to a new file.
echo "The username associated with this account is:`whoami`" >> garethduffy.txt
# who - Prints information about users who are currently logged in:
# Redirecting command output to a new file.
echo "The following users are logged in: `who`" >> garethduffy.txt
# finger - Displays information about the system users:
# Redirecting command output to a new file.
echo "Below is information pertaining to current users: `finger`" >> garethduffy.txt
# w - Shows who is logged on and what they are doing:
# Redirecting command output to a new file.
echo "The following users are logged in this is their activity `w`" >> garethduffy.txt
# history - Shows you all of the last commands that have been recently used:
```

```
# Redirecting command output to a new file.
echo "The following list commands have been recently used: `history`" >> garethduffy.txt
# echo | <enter>
# top - Displays Linux processes:
# Redirecting command output to a new file.
echo "The following processes are running: `top`" >> garethduffy.txt
# Gareth Duffy (G00364693) CATC, GMIT
# Assigment Q2.2; Shell script to automate execution of commands.
# This script will return the following set of information output and
# redirect each output to a text file named: garethduffy.txt
# To print the contents of this file in Linux type: cat garethduffy.txt
# date - Prints the current date and time:
# Redirecting command output to a new file.
echo "Today's date is: `date`" > garethduffy.txt
# hostname - Prints hostname information i.e. current IP address:
# Redirecting command output to a new file.
echo "The current IP address is: `hostname`" >> garethduffy.txt
# arch - Prints the machine kernel arcitecture info:
# Redirecting command output to a new file.
echo "The current machine architecture is: `arch`" >> garethduffy.txt
# uname -a - Prints all system information in a specific order:
# Redirecting command output to a new file.
echo "System information is `uname -a`" >> garethduffy.txt
# uptime - Tells the user how long the system has been running:
# Redirecting command output to a new file.
echo "This system has been running for: `uptime`" >> garethduffy.txt
# whoami - Prints  the  username  associated  with the current effective user ID:
# Redirecting command output to a new file.
echo "The username associated with this account is:`whoami`" >> garethduffy.txt
# who - Prints information about users who are currently logged in:
# Redirecting command output to a new file.
echo "The following users are logged in: `who`" >> garethduffy.txt
```

```
# finger - Displays information about the system users:
# Redirecting command output to a new file.
echo "Below is information pertaining to current users: `finger`" >> garethduffy.txt
# w - Shows who is logged on and what they are doing:
# Redirecting command output to a new file.
echo "The following users are logged in this is their activity `w`" >> garethduffy.txt
# history - Shows you all of the last commands that have been recently used:
# Redirecting command output to a new file.
echo "The following list commands have been recently used: `history`" >> garethduffy.txt
# echo | <enter>
# top - Displays Linux processes:
# Redirecting command output to a new file.
echo "The following processes are running: `top`" >> garethduffy.txt
cat garethduffy.txt
clear
echo -n > garethduffy.txt
cat garethduffy.txt
# Gareth Duffy (G00364693) CATC, GMIT
# Assigment Q2.2; Shell script to automate execution of commands.
# This script will return the following set of information output and
# redirect each output to a text file named: garethduffy.txt
# To print the contents of this file in Linux type: cat garethduffy.txt
# date - Prints the current date and time:
# Redirecting command output to a new file.
echo "Today's date is: `date`" > garethduffy.txt
# hostname - Prints hostname information i.e. current IP address:
# Redirecting command output to a new file.
echo "The current IP address is: `hostname`" >> garethduffy.txt
# arch - Prints the machine kernel arcitecture info:
# Redirecting command output to a new file.
echo "The current machine architecture is: `arch`" >> garethduffy.txt
# uname -a - Prints all system information in a specific order:
```

```
# Redirecting command output to a new file.
echo "System information is `uname -a`" >> garethduffy.txt
# uptime - Tells the user how long the system has been running:
# Redirecting command output to a new file.
echo "This system has been running for: `uptime`" >> garethduffy.txt
# whoami - Prints  the  username  associated  with the current effective user ID:
# Redirecting command output to a new file.
echo "The username associated with this account is:`whoami`" >> garethduffy.txt
# who - Prints information about users who are currently logged in:
# Redirecting command output to a new file.
echo "The following users are logged in: `who`" >> garethduffy.txt
# finger - Displays information about the system users:
# Redirecting command output to a new file.
echo "Below is information pertaining to current users: `finger`" >> garethduffy.txt
# w - Shows who is logged on and what they are doing:
# Redirecting command output to a new file.
echo "The following users are logged in this is their activity `w`" >> garethduffy.txt
# history - Shows you all of the last commands that have been recently used:
# Redirecting command output to a new file.
echo "The following list commands have been recently used: `history`" >> garethduffy.txt
echo | <enter>
# top - Displays Linux processes:
# Redirecting command output to a new file.
echo "The following processes are running: `top`" >> garethduffy.txt
cat garethduffy.txt
echo -n > garethduffy.txt
cat garethduffy.txt
# Gareth Duffy (G00364693) CATC, GMIT
# Assigment Q2.2; Shell script to automate execution of commands.
# This script will return the following set of information output and
# redirect each output to a text file named: garethduffy.txt
# To print the contents of this file in Linux type: cat garethduffy.txt
```

```
# date - Prints the current date and time:
# Redirecting command output to a new file.
echo "Today's date is: `date`" > garethduffy.txt
# hostname - Prints hostname information i.e. current IP address:
# Redirecting command output to a new file.
echo "The current IP address is: `hostname`" >> garethduffy.txt
# arch - Prints the machine kernel arcitecture info:
# Redirecting command output to a new file.
echo "The current machine architecture is: `arch`" >> garethduffy.txt
# uname -a - Prints all system information in a specific order:
# Redirecting command output to a new file.
echo "System information is `uname -a`" >> garethduffy.txt
# uptime - Tells the user how long the system has been running:
# Redirecting command output to a new file.
echo "This system has been running for: `uptime`" >> garethduffy.txt
# whoami - Prints  the  username  associated  with the current effective user ID:
# Redirecting command output to a new file.
echo "The username associated with this account is:`whoami`" >> garethduffy.txt
# who - Prints information about users who are currently logged in:
# Redirecting command output to a new file.
echo "The following users are logged in: `who`" >> garethduffy.txt
# finger - Displays information about the system users:
# Redirecting command output to a new file.
echo "Below is information pertaining to current users: `finger`" >> garethduffy.txt
# w - Shows who is logged on and what they are doing:
# Redirecting command output to a new file.
echo "The following users are logged in this is their activity `w`" >> garethduffy.txt
# history - Shows you all of the last commands that have been recently used:
# Redirecting command output to a new file.
echo "The following list commands have been recently used: `history`" >> garethduffy.txt
# echo | <enter>
# top - Displays Linux processes:
```

```
1630  # Redirecting command output to a new file.

1631  echo "The following processes are running: `top`" >> garethduffy.txt

1632  cat garethduffy.txt

1633  clear

1634  # Gareth Duffy (G00364693) GMIT CATC

1635  # The Mrs Doyle While Loop...Ah go on !

1636  # Shell script that behaves like an Irish person offering a cup of tea

1637  while true; do echo "Will you have a cup of tea?" ;    read yn;    if [[ $yn = y ]]; then       echo "Great, I'll
make tea!";       break;    elif [[ $yn = n ]]; then       echo "Are you sure?" ;       read yn;       if [[ $yn = n ]];
then          echo "Are you sure?";          read yn;          if [[ $yn = n ]]; then              echo "Are you sure?";
read yn;             if [[ $yn = n ]]; then                echo "Are you sure?";                read yn;                if [[ $yn
= n ]]; then                    break;                fi;             fi;          fi;       fi          ;    fi; done

1638  # Gareth Duffy (G00364693) GMIT CATC

1639  # The Mrs Doyle While Loop...Ah go on !

1640  # Shell script that behaves like an Irish person offering a cup of tea

1641  while true; do echo "Will you have a cup of tea?" ;    read yn;    if [[ $yn = y ]]; then       echo "Great, I'll
make tea!";       break;    elif [[ $yn = n ]]; then       echo "Are you sure?" ;       read yn;       if [[ $yn = n ]];
then          echo "Are you sure?";          read yn;          if [[ $yn = n ]]; then              echo "Are you sure?";
read yn;             if [[ $yn = n ]]; then                echo "Are you sure?";                read yn;                if [[ $yn
= n ]]; then                    break;                fi;             fi;          fi;       fi          ;    fi; done

1642  # Gareth Duffy (G00364693) CATC, GMIT

1643  # The Mrs Doyle While Loop...Ah go on !!

1644  # Shell script that behaves like an Irish person offering a cup of tea

1645  while true; do echo "Will you have a cup of tea?" ;    read yn;    if [[ $yn = y ]]; then       echo "Great, I'll
make tea!";       break;    elif [[ $yn = n ]]; then       echo "Are you sure?" ;       read yn;       if [[ $yn = n ]];
then          echo "Are you sure?";          read yn;          if [[ $yn = n ]]; then              echo "Are you sure?";
read yn;             if [[ $yn = n ]]; then                echo "Are you sure?";                read yn;                if [[ $yn
= n ]]; then                    break;                fi;             fi;          fi;       fi          ;    fi; done

1646  # Gareth Duffy (G00364693) CATC, GMIT

1647  # The Mrs Doyle While Loop...Ah go on !!

1648  # Shell script that behaves like an Irish person offering a cup of tea

1649  while true; do echo "Will you have a cup of tea?" ;    read yn;    if [[ $yn = y ]]; then       echo "Great, I'll
make tea!";       break;    elif [[ $yn = n ]]; then       echo "Are you sure?" ;       read yn;       if [[ $yn = n ]];
then          echo "Are you sure?";          read yn;          if [[ $yn = n ]]; then              echo "Are you sure?";
read yn;             if [[ $yn = n ]]; then                echo "Are you sure?";                read yn;                if [[ $yn
= n ]]; then                    break;                fi;             fi;          fi;       fi          ;    fi; done

1650  clear

1651  # Gareth Duffy (G00364693) CATC, GMIT
```

```
1652  # The Mrs Doyle While Loop...Ah go on !!

1653  # Shell script that behaves like an Irish person offering a cup of tea

1654  while true; do echo "Will you have a cup of tea?" ;     read yn;     if [[ $yn = y ]]; then        echo "Great, I'll make tea!";        break;     elif [[ $yn = n ]]; then        echo "Are you sure?" ;        read yn;        if [[ $yn = n ]]; then            echo "Are you sure?";            read yn;            if [[ $yn = n ]]; then                echo "Are you sure?";                read yn;                if [[ $yn = n ]]; then                    break;                fi;            fi;        fi;        fi        ;     fi; done

1655  echo -n > garethduffy.txt

1656  cat garethduffy.txt

1657  echo "Gareth Duffy (G00364693) CATC, GMIT" > garethduffy.txt

1658  echo "Assigment Q2.2; Shell script to automate execution of commands." >> garethduffy.txt

1659  echo "This script will output the following set of information and redirect each one to text file: garethduffy.txt" >> garethduffy.txt

1660  # date - Prints the current date and time:

1661  # Redirects command output to a new file.

1662  echo "Today's date is: `date`" >> garethduffy.txt

1663  # hostname - Prints hostname information i.e. current IP address:

1664  # Redirects command output to a new file.

1665  echo "The current IP address is: `hostname`" >> garethduffy.txt

1666  # arch - Prints the machine kernel arcitecture info:

1667  # Redirects command output to a new file.

1668  echo "The current machine architecture is: `arch`" >> garethduffy.txt

1669  # uname -a - Prints all system information in a specific order:

1670  # Redirects command output to a new file.

1671  echo "System information is `uname -a`" >> garethduffy.txt

1672  # uptime - Tells the user how long the system has been running:

1673  # Redirects command output to a new file.

1674  echo "This system has been running for: `uptime`" >> garethduffy.txt

1675  # whoami - Prints  the  username  associated  with the current effective user ID:

1676  # Redirects command output to a new file.

1677  echo "The username associated with this account is:`whoami`" >> garethduffy.txt

1678  # who - Prints information about users who are currently logged in:

1679  # Redirects command output to a new file.

1680  echo "The following users are logged in: `who`" >> garethduffy.txt
```

```
# finger - Displays information about the system users:
# Redirects command output to a new file.
echo "Below is information pertaining to current users: `finger`" >> garethduffy.txt
# w - Shows who is logged on and what they are doing:
# Redirects command output to a new file.
echo "The following users are logged in this is their activity `w`" >> garethduffy.txt
# history - Shows you all of the last commands that have been recently used:
# Redirects command output to a new file.
echo "The following list commands have been recently used: `history`" >> garethduffy.txt
```

The following processes are running:

```
[?1h=[?25l[H[2J(B[mtop - 19:53:47 up 45 days, 20:23, 6 users, load average: 0.00, 0.00, 0.00(B[m[39;49m(B[m[39;49m[K

Tasks:(B[m[39;49m[1m 137 (B[m[39;49mtotal,(B[m[39;49m[1m 1 (B[m[39;49mrunning,(B[m[39;49m[1m 136 (B[m[39;49msleeping,(B[m[39;49m[1m 0 (B[m[39;49mstopped,(B[m[39;49m[1m 0 (B[m[39;49mzombie(B[m[39;49m(B[m[39;49m[K

%Cpu(s):(B[m[39;49m[1m 0.1 (B[m[39;49mus,(B[m[39;49m[1m 0.0 (B[m[39;49msy,(B[m[39;49m[1m 0.0 (B[m[39;49mni,(B[m[39;49m[1m 99.9 (B[m[39;49mid,(B[m[39;49m[1m 0.0 (B[m[39;49mwa,(B[m[39;49m[1m 0.0 (B[m[39;49mhi,(B[m[39;49m[1m 0.0 (B[m[39;49msi,(B[m[39;49m[1m 0.0 (B[m[39;49mst(B[m[39;49m(B[m[39;49m[K

KiB Mem :(B[m[39;49m[1m 1014552 (B[m[39;49mtotal,(B[m[39;49m[1m 91728 (B[m[39;49mfree,(B[m[39;49m[1m 87072 (B[m[39;49mused,(B[m[39;49m[1m 835752 (B[m[39;49mbuff/cache(B[m[39;49m(B[m[39;49m[K

KiB Swap:(B[m[39;49m[1m 0 (B[m[39;49mtotal,(B[m[39;49m[1m 0 (B[m[39;49mfree,(B[m[39;49m[1m 0 (B[m[39;49mused.(B[m[39;49m[1m 704600 (B[m[39;49mavail Mem (B[m[39;49m(B[m[39;49m[K

[K

[7m PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND (B[m[39;49m[K

(B[m 1 root 20 0 119820 5392 3456 S 0.0 0.5 0:38.49 systemd (B[m[39;49m[K

(B[m 2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kthreadd (B[m[39;49m[K

(B[m 3 root 20 0 0 0 0 S 0.0 0.0 0:15.64 ksoftirqd/0 (B[m[39;49m[K

(B[m 5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H (B[m[39;49m[K
```

(B[m 7 root 20 0 0 0 0 S 0.0 0.0 0:27.28 rcu_sched
(B[m[39;49m[K

(B[m 8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
(B[m[39;49m[K

(B[m 9 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
(B[m[39;49m[K

(B[m 10 root rt 0 0 0 0 S 0.0 0.0 0:18.24 watchdog/0
(B[m[39;49m[K

(B[m 11 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
(B[m[39;49m[K

(B[m 12 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 netns
(B[m[39;49m[K

(B[m 13 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 perf
(B[m[39;49m[K

(B[m 14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenwatch
(B[m[39;49m[K

(B[m 15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenbus
(B[m[39;49m[K

(B[m 17 root 20 0 0 0 0 S 0.0 0.0 0:00.76 khungtaskd
(B[m[39;49m[K

(B[m 18 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 writeback
(B[m[39;49m[K

(B[m 19 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksmd
(B[m[39;49m[K

(B[m 20 root 39 19 0 0 0 S 0.0 0.0 0:06.98 khugepaged
(B[m[39;49m[K

(B[m 21 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 crypto
(B[m[39;49m[K

(B[m 22 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kintegrityd
(B[m[39;49m[K

(B[m 23 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 24 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kblockd
(B[m[39;49m[K

(B[m 25 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 ata_sff
(B[m[39;49m[K

(B[m 26 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 md
(B[m[39;49m[K

(B[m 27 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 devfreq_wq
(B[m[39;49m[K

(B[m 30 root 20 0 0 0 0 S 0.0 0.0 0:00.47 kswapd0
(B[m[39;49m[K

(B[m 31 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 vmstat
(B[m[39;49m[K

(B[m 32 root 20 0 0 0 0 S 0.0 0.0 0:00.00 fsnotify_mark
(B[m[39;49m[K

(B[m 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ecryptfs-kthrea
(B[m[39;49m[K

(B[m 49 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kthrotld
(B[m[39;49m[K

(B[m 50 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 51 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 52 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 53 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 54 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 55 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 56 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 57 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 58 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 59 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K[H(B[mtop - 19:53:50 up 45 days, 20:23, 6 users, load average: 0.00, 0.00, 0.00(B[m[39;49m(B[m[39;49m[K

Tasks:(B[m[39;49m[1m 137 (B[m[39;49mtotal,(B[m[39;49m[1m 1 (B[m[39;49mrunning,(B[m[39;49m[1m 136 (B[m[39;49msleeping,(B[m[39;49m[1m 0 (B[m[39;49mstopped,(B[m[39;49m[1m 0 (B[m[39;49mzombie(B[m[39;49m(B[m[39;49m[K

%Cpu(s):(B[m[39;49m[1m 0.0 (B[m[39;49mus,(B[m[39;49m[1m 0.0 (B[m[39;49msy,(B[m[39;49m[1m 0.0 (B[m[39;49mni,(B[m[39;49m[1m100.0 (B[m[39;49mid,(B[m[39;49m[1m 0.0

(B[m[39;49mwa,(B[m[39;49m[1m 0.0 (B[m[39;49mhi,(B[m[39;49m[1m 0.0 (B[m[39;49msi,(B[m[39;49m[1m 0.0 (B[m[39;49mst(B[m[39;49m(B[m[39;49m[K

KiB Mem :(B[m[39;49m[1m 1014552 (B[m[39;49mtotal,(B[m[39;49m[1m 91728 (B[m[39;49mfree,(B[m[39;49m[1m 87064 (B[m[39;49mused,(B[m[39;49m[1m 835760 (B[m[39;49mbuff/cache(B[m[39;49m(B[m[39;49m[K

[K

[?25l[H(B[mtop - 19:53:51 up 45 days, 20:23, 6 users, load average: 0.00, 0.00, 0.00(B[m[39;49m(B[m[39;49m[K

Tasks:(B[m[39;49m[1m 137 (B[m[39;49mtotal,(B[m[39;49m[1m 1 (B[m[39;49mrunning,(B[m[39;49m[1m 136 (B[m[39;49msleeping,(B[m[39;49m[1m 0 (B[m[39;49mstopped,(B[m[39;49m[1m 0 (B[m[39;49mzombie(B[m[39;49m(B[m[39;49m[K

%Cpu(s):(B[m[39;49m[1m 0.0 (B[m[39;49mus,(B[m[39;49m[1m 0.0 (B[m[39;49msy,(B[m[39;49m[1m 0.0 (B[m[39;49mni,(B[m[39;49m[1m100.0 (B[m[39;49mid,(B[m[39;49m[1m 0.0 (B[m[39;49mwa,(B[m[39;49m[1m 0.0 (B[m[39;49mhi,(B[m[39;49m[1m 0.0 (B[m[39;49msi,(B[m[39;49m[1m 0.0 (B[m[39;49mst(B[m[39;49m(B[m[39;49m[K

KiB Mem :(B[m[39;49m[1m 1014552 (B[m[39;49mtotal,(B[m[39;49m[1m 91728 (B[m[39;49mfree,(B[m[39;49m[1m 87064 (B[m[39;49mused,(B[m[39;49m[1m 835760 (B[m[39;49mbuff/cache(B[m[39;49m(B[m[39;49m[K

KiB Swap:(B[m[39;49m[1m 0 (B[m[39;49mtotal,(B[m[39;49m[1m 0 (B[m[39;49mfree,(B[m[39;49m[1m 0 (B[m[39;49mused.(B[m[39;49m[1m 704600 (B[m[39;49mavail Mem (B[m[39;49m(B[m[39;49m[K

[K

[7m PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND (B[m[39;49m[K

(B[m 1 root 20 0 119820 5392 3456 S 0.0 0.5 0:38.49 systemd (B[m[39;49m[K

(B[m 2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kthreadd (B[m[39;49m[K

(B[m 3 root 20 0 0 0 0 S 0.0 0.0 0:15.64 ksoftirqd/0 (B[m[39;49m[K

(B[m 5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H (B[m[39;49m[K

(B[m 7 root 20 0 0 0 0 S 0.0 0.0 0:27.28 rcu_sched (B[m[39;49m[K

(B[m 8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh (B[m[39;49m[K

(B[m 9 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0 (B[m[39;49m[K

```
(B[m   10 root     rt  0     0     0     0 S  0.0  0.0   0:18.24 watchdog/0
(B[m[39;49m[K
(B[m   11 root     20  0     0     0     0 S  0.0  0.0   0:00.00 kdevtmpfs
(B[m[39;49m[K
(B[m   12 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 netns
(B[m[39;49m[K
(B[m   13 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 perf
(B[m[39;49m[K
(B[m   14 root     20  0     0     0     0 S  0.0  0.0   0:00.00 xenwatch
(B[m[39;49m[K
(B[m   15 root     20  0     0     0     0 S  0.0  0.0   0:00.00 xenbus
(B[m[39;49m[K
(B[m   17 root     20  0     0     0     0 S  0.0  0.0   0:00.76 khungtaskd
(B[m[39;49m[K
(B[m   18 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 writeback
(B[m[39;49m[K
(B[m   19 root     25  5     0     0     0 S  0.0  0.0   0:00.00 ksmd
(B[m[39;49m[K
(B[m   20 root     39 19     0     0     0 S  0.0  0.0   0:06.98 khugepaged
(B[m[39;49m[K
(B[m   21 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 crypto
(B[m[39;49m[K
(B[m   22 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 kintegrityd
(B[m[39;49m[K
(B[m   23 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 bioset
(B[m[39;49m[K
(B[m   24 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 kblockd
(B[m[39;49m[K
(B[m   25 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 ata_sff
(B[m[39;49m[K
(B[m   26 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 md
(B[m[39;49m[K
(B[m   27 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 devfreq_wq
(B[m[39;49m[K
(B[m   30 root     20  0     0     0     0 S  0.0  0.0   0:00.47 kswapd0
(B[m[39;49m[K
(B[m   31 root      0 -20    0     0     0 S  0.0  0.0   0:00.00 vmstat
(B[m[39;49m[K
```

(B[m 32 root 20 0 0 0 0 S 0.0 0.0 0:00.00 fsnotify_mark
(B[m[39;49m[K

(B[m 33 root 20 0 0 0 0 S 0.0 0.0 0:00.00 ecryptfs-kthrea
(B[m[39;49m[K

(B[m 49 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kthrotld
(B[m[39;49m[K

(B[m 50 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 51 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 52 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 53 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 54 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 55 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 56 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 57 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 58 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K

(B[m 59 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset
(B[m[39;49m[K[?1l>[47;1H

[?12l[?25h[K

Q2.3

Q2.3.1

Below are screenshots of the Linux access permissions changed so that my personal folder became fully accessible to myself alone, but completely inaccessible to the group and all others:

```
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ ls -l
total 32
-rw-rw-r-- 1 garethduffy garethduffy 25110 Apr 14 14:57 garethduffy.txt
drwxrwxr-x 5 garethduffy garethduffy  4096 Mar 20 17:23 Music
```

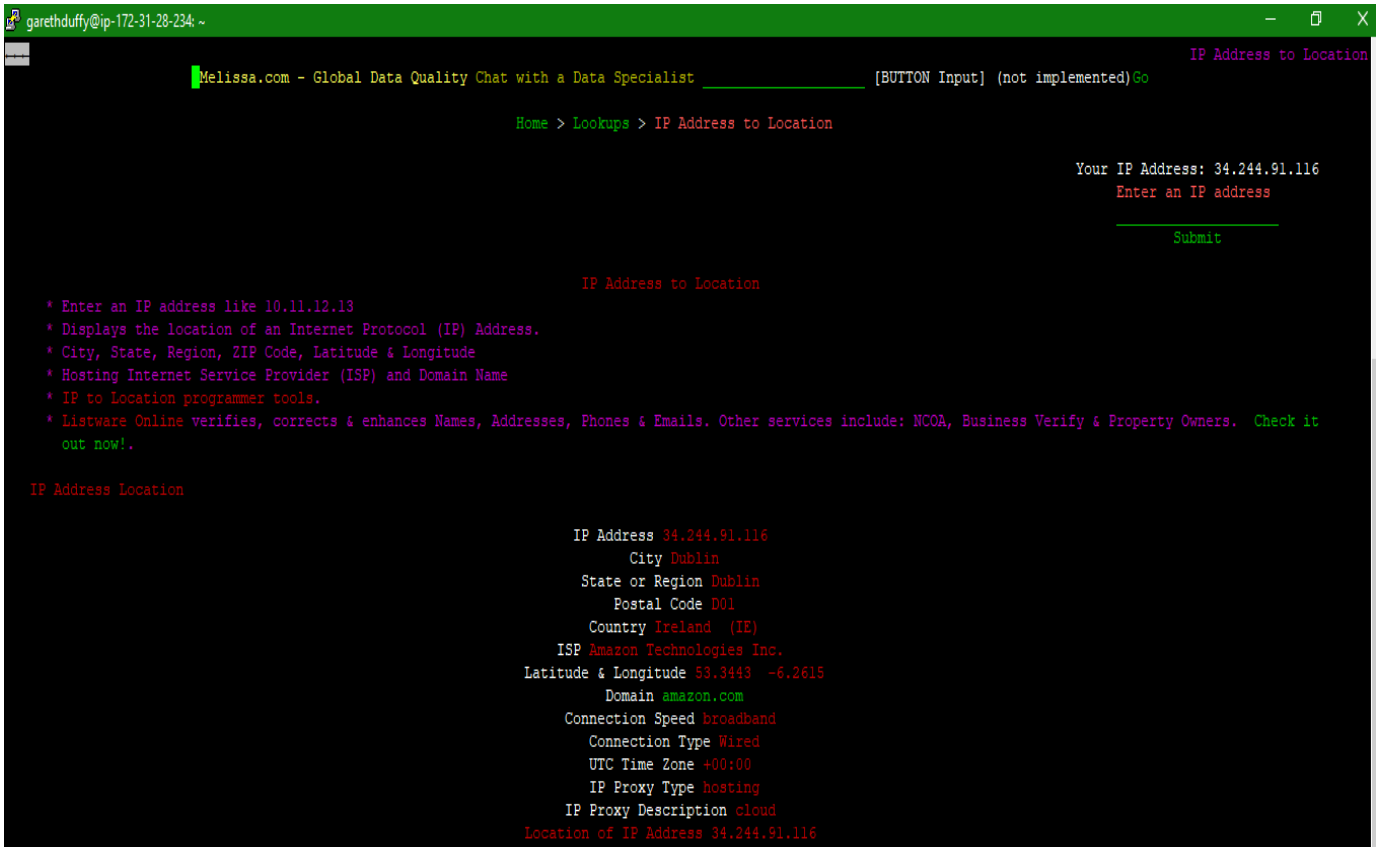
Access permissions *before* changes

```
garethduffy@ip-172-31-28-234:~$ ls -l
total 32
-rwx------ 1 garethduffy garethduffy 25110 Apr 14 14:57 garethduffy.txt
drwx------ 5 garethduffy garethduffy  4096 Mar 20 17:23 Music
garethduffy@ip-172-31-28-234:~$
```

Access permissions *after* changes

Q2.3.2

Below is a screenshot of the IP location service I used to determine the city and country (Dublin, Ireland) where the VM is located (via Lynx the browser):



Q2.4

Shell script program that behaves like an Irish person offering a cup of tea.

The screenshots below show the shell script in operation with the three possible outcomes [2]:

-

```
garethduffy@ip-172-31-28-234: ~
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

69 packages can be updated.
0 updates are security updates.


*** System restart required ***
Last login: Sat Apr 28 18:17:19 2018 from 37.228.242.22
garethduffy@ip-172-31-28-234:~$ #!/bin/sh
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ # Gareth Duffy (G00364693) CATC, GMIT
garethduffy@ip-172-31-28-234:~$ # The Mrs Doyle While Loop...Ah go on !!
garethduffy@ip-172-31-28-234:~$ # Shell script that behaves like an Irish person offering a cup of tea
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ read -p "Will you have a cup of tea?" yn; while true; do
>
>     if [[ $yn = y ]]; then
>         echo "Great, I'll make tea!"
>         break
>     elif [[ $yn = n ]]; then
>         echo "Are you sure?"
>         read yn
>         if [[ $yn = n ]]; then
>             echo "Are you sure?"
>             read yn
>             if [[ $yn = n ]]; then
>                 echo "Are you sure?"
>                 read yn
>                 if [[ $yn = n ]]; then
>                     echo "Are you sure?"
>                     read yn
>                     if [[ $yn = n ]]; then
>                         break
>                     fi
>                 fi
>             fi
>         fi
>     fi
> done
Will you have a cup of tea? y
Great, I'll make tea!
garethduffy@ip-172-31-28-234:~$
```

Screenshot of user typing “y” (yes) immediately after first tea offer.

```
garethduffy@ip-172-31-28-234: ~
garethduffy@ip-172-31-28-234:~$ #!/bin/sh
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ # Gareth Duffy (G00364693) CATC, GMIT
garethduffy@ip-172-31-28-234:~$ # The Mrs Doyle While Loop...Ah go on !!
garethduffy@ip-172-31-28-234:~$ # Shell script that behaves like an Irish person offering a cup of tea
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ read -p "Will you have a cup of tea?" yn; while true; do
>
>     if [[ $yn = y ]]; then
>         echo "Great, I'll make tea!"
>         break
>     elif [[ $yn = n ]]; then
>         echo "Are you sure?"
>         read yn
>         if [[ $yn = n ]]; then
>             echo "Are you sure?"
>             read yn
>             if [[ $yn = n ]]; then
>                 echo "Are you sure?"
>                 read yn
>                 if [[ $yn = n ]]; then
>                     echo "Are you sure?"
>                     read yn
>                     if [[ $yn = n ]]; then
>                         break
>                     fi
>                 fi
>             fi
>         fi
>     fi
> done
Will you have a cup of tea? n
Are you sure?
n
Are you sure?
n
Are you sure?
n
Are you sure?
n
garethduffy@ip-172-31-28-234:~$
```

Screenshot of user typing “n” (no) four consecutive times before Mrs Doyle gives up.

```
garethduffy@ip-172-31-28-234: ~
garethduffy@ip-172-31-28-234:~$ #!/bin/sh
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ # Gareth Duffy (G00364693) CATC, GMIT
garethduffy@ip-172-31-28-234:~$ # The Mrs Doyle While Loop...Ah go on !!
garethduffy@ip-172-31-28-234:~$ # Shell script that behaves like an Irish person offering a cup of tea
garethduffy@ip-172-31-28-234:~$
garethduffy@ip-172-31-28-234:~$ read -p "Will you have a cup of tea?" yn; while true; do
>
>     if [[ $yn = y ]]; then
>         echo "Great, I'll make tea!"
>         break
>     elif [[ $yn = n ]]; then
>         echo "Are you sure?"
>         read yn
>         if [[ $yn = n ]]; then
>             echo "Are you sure?"
>             read yn
>             if [[ $yn = n ]]; then
>                 echo "Are you sure?"
>                 read yn
>                 if [[ $yn = n ]]; then
>                     echo "Are you sure?"
>                     read yn
>                     if [[ $yn = n ]]; then
>                         break
>                     fi
>                 fi
>             fi
>         fi
>     fi
> done
Will you have a cup of tea? n
Are you sure?
n
Are you sure?
n
Are you sure?
y
Great, I'll make tea!
garethduffy@ip-172-31-28-234:~$
```

Screenshot of user first refusing the first three tea offers but saying yes on the fourth offer.

Shell script code:

```
#!/bin/sh

# Gareth Duffy (G00364693) CATC, GMIT
# The Mrs Doyle While Loop...Ah go on !!
# Shell script that behaves like an Irish person offering a cup of tea

read -p "Will you have a cup of tea?" yn; while true; do
```

```
    if [[ $yn = y ]]; then
        echo "Great, I'll make tea!"
        break
    elif [[ $yn = n ]]; then
        echo "Are you sure?"
        read yn
        if [[ $yn = n ]]; then
            echo "Are you sure?"
            read yn
            if [[ $yn = n ]]; then
                echo "Are you sure?"
                read yn
                if [[ $yn = n ]]; then
                    break
                fi
            fi
        fi
    fi
    fi
done
```

References:

[1]. Author unknown, (2017). *Binary Adder and Subtractor*. Retrieved from: https://www.electronicshub.org/binary-adder-and-subtractor/

[2]. Chadwick. R. (2018). *Bash Scripting Tutorial*. Retrieved from: https://ryanstutorials.net/bash-scripting-tutorial/bash-loops.php

[3]. Author unknown (2018). *Circuit Lab Workbench.* Retrieved from: https://www.circuitlab.com/