Operating Systems Assignment

Practical Report

Zalán Tóth - 20102768
Computer Forensics and Security Year 2

SETU Waterford
School of Science and Computing
Department of Computing and Mathematics

Waterford City, Ireland
17/12/2023

This document was made with Google Documents, therefore I'm attaching a PDF version as well alongside the DOCX for better readability!

Table of Contents

Practicals 1-5

1. Exploring the use of commands with options and arguments

Taking a selection of Windows CLI commands from those given below, use the online help to examine the various options and arguments, and try them out. You can look up others at: http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/enus/ntcmds.mspx?mfr=true You're required carefully to write two A4 pages (Times 12 point or equivalent size) detailing your experiments with different options for between six and ten different commands. To get the online help for a command, type command /? 6. e.g. dir /? prompt mkdir color title tree type ver print xcopy Type help at the windows command line prompt to see some more instructions

Answer:

1. dir
   This command displays a list of files and subdirectories in a directory.
   With option *A* you're able to display files with specific attributes, but you have to specify the attribute. There are 8 different ones and an extra (-) which is the negation for those 8 attributes. This means you can specify things like directories (D), hidden files (H), system files (S), reparse points (L), read-only files (R), archive-ready files (A), not content indexed files (I), offline files (O). That means if I type in dir /A:D it will only output the subdirectories in the directory. Or if we wanna see everything except directories, you can type in dir /A:-D as the - stands for negation, which reverses the function of the attribute.
2. prompt
   This command changes the command prompt of the terminal (cmd.exe).
   You can add many specific placeholders to the prompt. The default one is your current location in the directory hierarchy. With the placeholder $D you're entering the current date or $T for the current time. $N provides the current drive, which is C:\ for me. $P gives us the current drive and path. We're able to use space with $S or if we need the dollar ($) symbol we have to use the $$ placeholder. For some special characters, we have to use specific placeholders. To set it to the default one we only have to enter the command "prompt $P$G". The $P gives the path and the $G gives the special symbol >. We can also add the Windows version number with $V, which gives the prompt "Microsoft Windows [Version 10.0.22621.2134]" for my PC.
3. mkdir
   This command creates a directory with the given name.
   You can also use md as a shortcut. You have to provide the relative path or directly the drive and the path. This means if I type in "mkdir testdir" it will create a directory called testdir in the directory we're currently in. If I type in "mkdir

C:\Users\Zalan\Projects\SETU_Semester_3\hello" it will create a subdirectory called hello inside SETU_Semester_3 in drive C as can be seen in the command.

4. color
   This command sets the default console foreground and background colors.
   The attributes for this command are simple, we only have to use two hex digits. The first corresponds to the background and the second one to the foreground. As it can be given in hex, we have 16 options for colors, like 0 for black and F for white. This means if I type in "color 0F", it'll set the console background to black and the text (foreground) to white. If I type in "color 1F", it'll set the background to blue and the foreground to white.
5. title
   This command sets the window title for the terminal window.
   Its only attribute is a string, which specifies the title for the window. So if I type in "title Hello" it'll change the window's name to Hello.
6. tree
   This command graphically displays the directory hierarchy of the given drive/path.
   We can specify the path in the [drive:][path] format, or just leave it out, and it'll display the current working directory. It can also display the filenames in each folder with option /F. So if we wanna list out every folder and file in drive C, the command would be "tree C:\ /F".
7. type
   This command displays the contents of a file.
   We can specify the drive and path to the file, as well as just look for the file in the current working directory. If we have the file hello.txt in our directory, the command to display it would be "type hello.txt".
8. ver
   This command displays the version of Windows. There are no attributes for this command.
9. print
   This command prints a text file.
10. xcopy
    This command copies files and directory trees.
    We can use it with the syntax "XCOPY [source] [destination] [attributes]". There are many attributes for this command. You can for example ask for a prompt before creating each destination file with the /P attribute. Or if you don't wanna display file names while copying, we can use /Q, or if we need that we can use the /L attribute. For even more information /F displays full source and destination file names while copying. /H also copies hidden and system files as well. You can also specify a date and only files on or after that date will be copied (/D:m-d-y).

2. What's the purpose of the first line - @ECHO OFF? Remove it and see the effect.

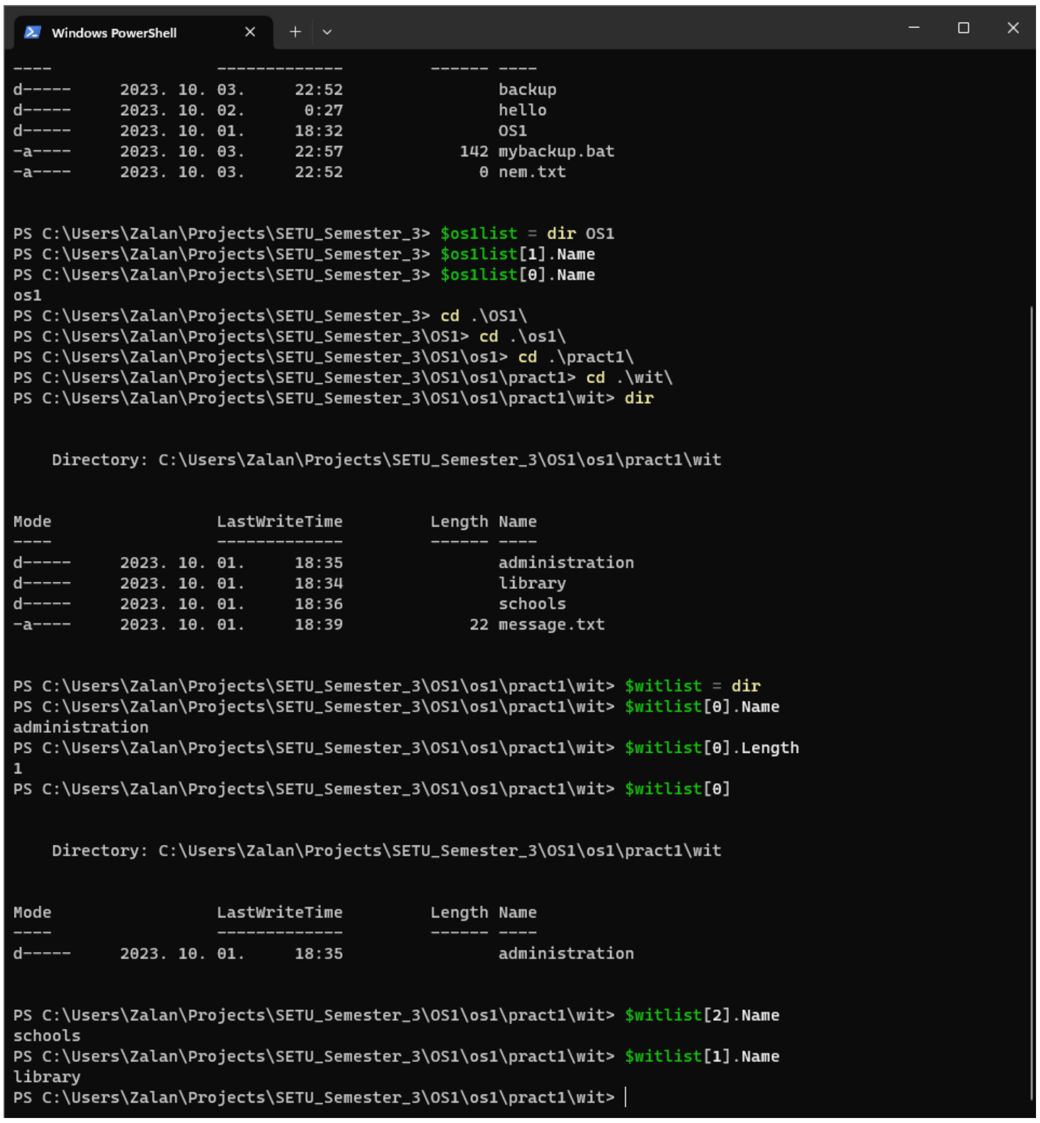
```
@ECHO OFF
ECHO please insert a usb memory stick
PAUSE
COPY *.txt I:\
ECHO BACKUP COMPLETE
```

Answer:

It prevents the prompt and the content of the file from displaying, which creates a more user-friendly experience.

3. Test the above using a directory that you created last week.

Answer:



```
----              -------------        ------ ----
d-----     2023. 10. 03.     22:52                backup
d-----     2023. 10. 02.      0:27                hello
d-----     2023. 10. 01.     18:32                OS1
-a----     2023. 10. 03.     22:57            142 mybackup.bat
-a----     2023. 10. 03.     22:52              0 nem.txt


PS C:\Users\Zalan\Projects\SETU_Semester_3> $os1list = dir OS1
PS C:\Users\Zalan\Projects\SETU_Semester_3> $os1list[1].Name
PS C:\Users\Zalan\Projects\SETU_Semester_3> $os1list[0].Name
os1
PS C:\Users\Zalan\Projects\SETU_Semester_3> cd .\OS1\
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1> cd .\os1\
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1> cd .\pract1\
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1> cd .\wit\
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> dir


    Directory: C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----     2023. 10. 01.     18:35                administration
d-----     2023. 10. 01.     18:34                library
d-----     2023. 10. 01.     18:36                schools
-a----     2023. 10. 01.     18:39             22 message.txt


PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> $witlist = dir
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> $witlist[0].Name
administration
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> $witlist[0].Length
1
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> $witlist[0]


    Directory: C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----     2023. 10. 01.     18:35                administration


PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> $witlist[2].Name
schools
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> $witlist[1].Name
library
PS C:\Users\Zalan\Projects\SETU_Semester_3\OS1\os1\pract1\wit> |
```

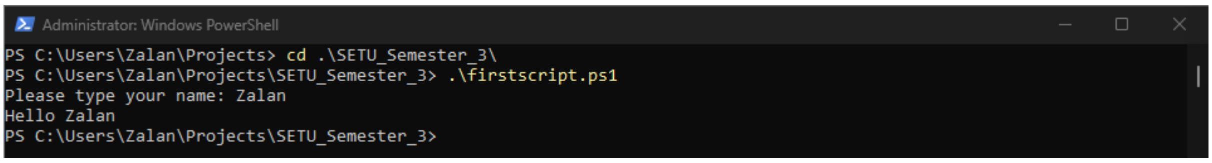
---

4. Simply record what happens when you run this script.

```
$name = Read-Host "Please type your name"
Write-Host "Hello" $name
```

Answer:

This script simply prompts for a string and then writes it out, as can be seen below:



```
Administrator: Windows PowerShell
PS C:\Users\Zalan\Projects> cd .\SETU_Semester_3\
PS C:\Users\Zalan\Projects\SETU_Semester_3> .\firstscript.ps1
Please type your name: Zalan
Hello Zalan
PS C:\Users\Zalan\Projects\SETU_Semester_3>
```

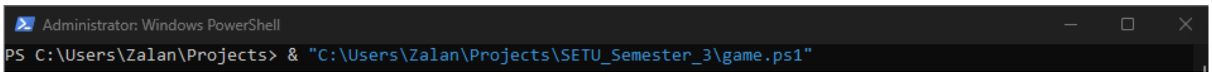
---

5. Find out how to run scripts if they're in a directory other than the working directory.

Answer:

We have to use the call operator for that purpose, which can be called with the & symbol.

So the example for running my script on an absolute route would be:



```
Administrator: Windows PowerShell
PS C:\Users\Zalan\Projects> & "C:\Users\Zalan\Projects\SETU_Semester_3\game.ps1"
```

---

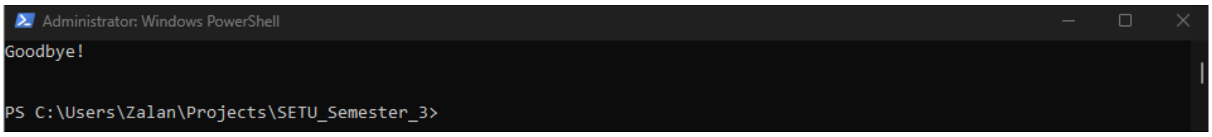
6. Study the following script and see if you can figure out what it'll do. Now type it into a file called game.ps1 (You can use copy/paste in places to reduce the labour.) Run it and see if your predictions are true

```
############################################################
############################################################
##
## The three pathetic knock-knock jokes program!
## Date: 26/09/17
## For: BSc (Hons) Computer Forensics and Security
##
############################################################
############################################################
############################
## initialisation section
############################
$userReply = ""
```

```
###########################
## first question
###########################
Clear-Host
14.
while($userReply -ne "Who is there?"){
$userReply = read-host "Knock Knock!"
}
Clear-Host
while($userReply -ne "Orange who?"){
$userReply = read-host "Orange"
}
Clear-Host
Write-Output "Orange you glad you created this PowerShell script?"
Start-Sleep -Seconds 5
###########################
## Second Question
###########################
Clear-Host
while($userReply -ne "Who is there?"){
$userReply = read-host "Knock Knock!"
}
Clear-Host
while($userReply -ne "Orange who?"){
$userReply = read-host "Orange"
}
Clear-Host
Write-Output "Oranges are oranges but this is PowerShellscripting!"
Start-Sleep -Seconds 5
###########################
## Third Question
###########################
Clear-Host
while($userReply -ne "Who is there?"){
$userReply = read-host "Knock Knock!"
}
15.
Clear-Host
while($userReply -ne "Banana who?"){
$userReply = read-host "Banana"
}
Clear-Host
Write-Output "Orange you glad I didn't say orange?"
Start-Sleep -Seconds 5
############################
## Farewell Message
############################
Clear-Host
Write-Output "Goodbye!`n`n"
```

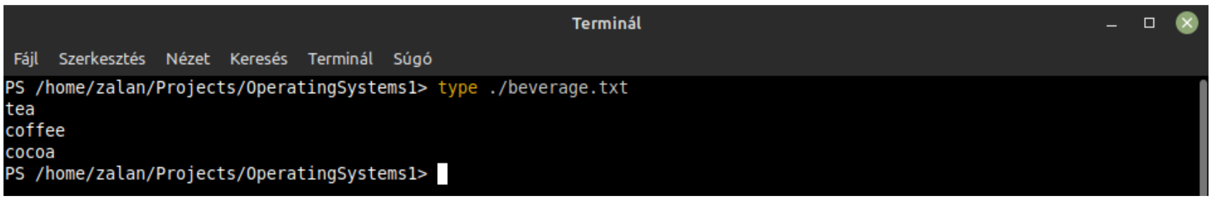
Answer:

It's a simple knock-knock game. I have to answer/question the correct string, so I can move on with the game.



```
Goodbye!

PS C:\Users\Zalan\Projects\SETU_Semester_3>
```

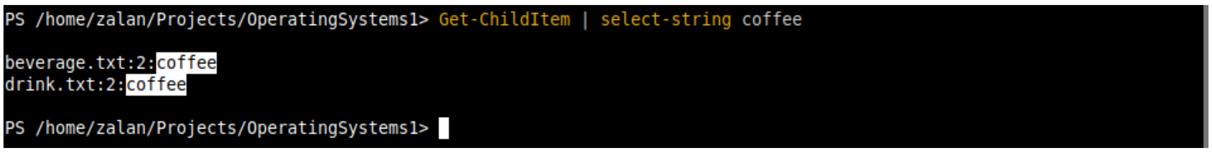
7. Create another file called beverage.txt identical to drink.txt



```
PS /home/zalan/Projects/OperatingSystems1> type ./beverage.txt
tea
coffee
cocoa
PS /home/zalan/Projects/OperatingSystems1>
```

8. Issue this command again

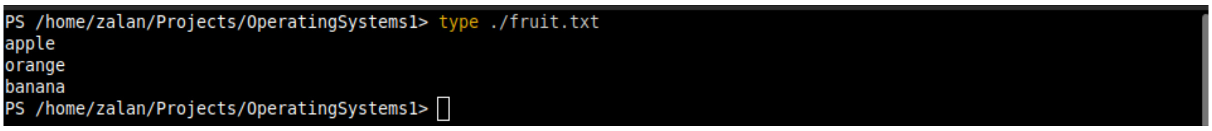
PS C:\Users\User1\green> Get-ChildItem | select-string coffee and see what happens.



```
PS /home/zalan/Projects/OperatingSystems1> Get-ChildItem | select-string coffee

beverage.txt:2:coffee
drink.txt:2:coffee

PS /home/zalan/Projects/OperatingSystems1>
```

It shows every file that contains the string coffee in the directory.

9. Create a file called fruit.txt with the list apple orange banana in it.

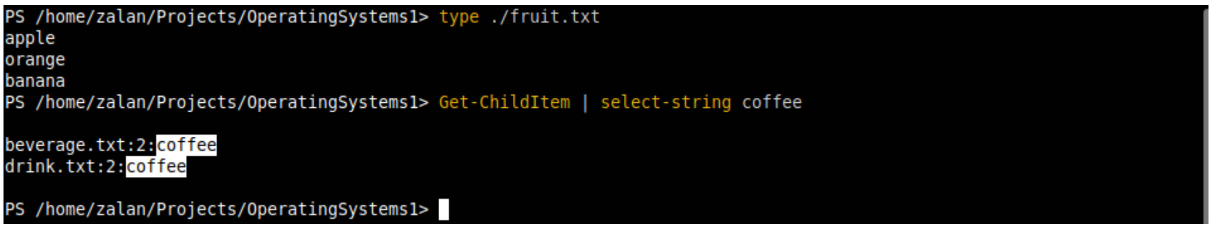


```
PS /home/zalan/Projects/OperatingSystems1> type ./fruit.txt
apple
orange
banana
PS /home/zalan/Projects/OperatingSystems1> 
```

10. Issue this command again

PS C:\Users\User1\green> Get-ChildItem | select-string coffee and see what happens.

It shows the same as before, as the newly created fruit.txt does not contain the string coffee.



```
PS /home/zalan/Projects/OperatingSystems1> type ./fruit.txt
apple
orange
banana
PS /home/zalan/Projects/OperatingSystems1> Get-ChildItem | select-string coffee

beverage.txt:2:coffee
drink.txt:2:coffee

PS /home/zalan/Projects/OperatingSystems1> 
```

11. Find out and explain how to get help about any cmdlet

We type in Get-Help and then the command we wanna get help with, so for example: “Get-Help Get-ChildItem”.

12. Find out and explain what F7 does in PowerShell

It does display the command history executed in the current section. This allows to quickly search, and navigate through previously run commands and even execute them without having to retype them.

13. What is the purposes of (a) the –whatif switch and (b) the –confirm

a) When using the -WhatIf attribute, PowerShell simulates the execution of the program, so we can preview changes without making them. This can be a very useful attribute, especially for testing scripts.
b) Using the -Confirm attribute, PowerShell will prompt the user for permission before executing the next step. It acts as a safe tool, as users have to authenticate the actions to be run.

14. Write a note to explain how you can use tab to complete a command as soon as it's unambiguous.

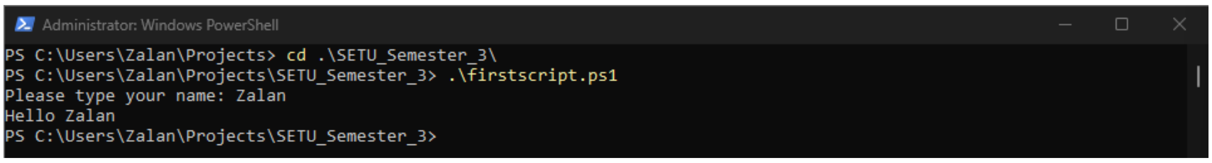
Using the Tab key allows us to use autocomplete for commands. It can complete parameters, paths, etc. It speeds up doing anything, as you can write commands faster. It'll only finish the parameter completely if it is unambiguous and matches only one item.

15. Simply record what happens when you run this script. What difference does it make if you leave out the text "Please type your name" from the first line of the script?

```
$name = Read-Host "Please type your name"

Write-Host "Hello" $name
```

Answer:

This script simply prompts for a string and then writes it out, as can be seen below:

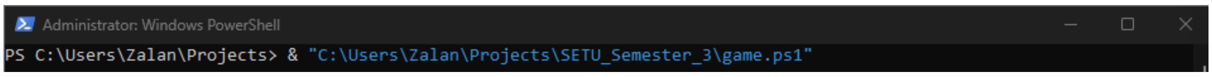


```
PS C:\Users\Zalan\Projects> cd .\SETU_Semester_3\
PS C:\Users\Zalan\Projects\SETU_Semester_3> .\firstscript.ps1
Please type your name: Zalan
Hello Zalan
PS C:\Users\Zalan\Projects\SETU_Semester_3>
```

16. Find out how to run scripts if they're in a directory other than the working directory.

We have to use the call operator for that purpose, which can be called with the & symbol.

So the example for running my script on an absolute route would be:



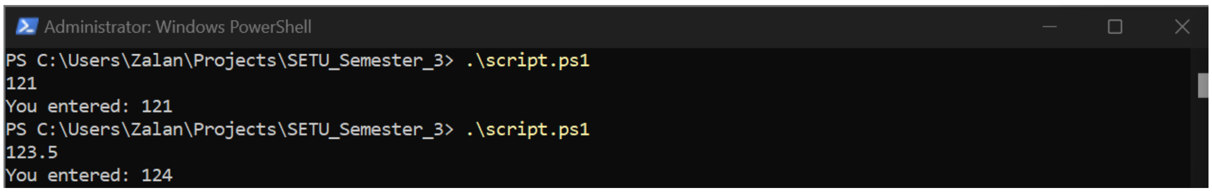
```
PS C:\Users\Zalan\Projects> & "C:\Users\Zalan\Projects\SETU_Semester_3\game.ps1"
```

17. (a) Find out how to do the same thing that the following code does except that it'll only accept integers (such as 67). (b) Once you've found the answer, find out what happens if you type a real number (such as 67.4 or 67.8).

We have to replace the datatype Double with Int. If we input a double, it will round to the closest integer.

```
$inputString = read-host
```

```
$value = $inputString -as [Double]

write-host "You entered: $value"
```



```
Administrator: Windows PowerShell
PS C:\Users\Zalan\Projects\SETU_Semester_3> .\script.ps1
121
You entered: 121
PS C:\Users\Zalan\Projects\SETU_Semester_3> .\script.ps1
123.5
You entered: 124
```

18. Alter the following program to require the user to enter specifically integer values between 1 and 4 inclusive.

Original:

```
do

{

 write-host -nonewline "Enter a numeric value: "

 $inputString = read-host

 $value = $inputString -as [Double]

 $ok = $value -ne $NULL

 if ( -not $ok ) { write-host "You must enter a numeric value" }

}

until ( $ok )

write-host "You entered: $value"
```

Rewritten:

```
do

{

 write-host -nonewline "Enter a numeric value: "

 $inputString = read-host

 $value = $inputString -as [Double]

 $between = $value

 $ok = ($between -ge 1) -and ($between -le 4)

 if ( -not $ok ) { write-host "You must enter a number between 1 and 4
inclusive" }

}

until ( $ok )

write-host "You entered: $value"
```

19. What does the following script do? (shell while)

It prints out numbers from 1 to 20 into the terminal, excluding 13 and 17.

```
$i = 1
while ($i -le 20)
{
if(($i -ne 13) -and ($i -ne 17))
{
Write-Host $i
}
$i = $i + 1
}
```

20. We don't need the brackets round ($i -ne 13) and ($i -ne 17) in the code above. Why do we not need them? (Hint: The answer is the same as for Java). Do you think that it's a good idea to put them in even if they aren't necessary? Explain your answer.

Because it creates one condition altogether. I however it's a good idea to use them as they provide a clean overlook when looking into the script.

21. We can replace $i = $i + 1 with something shorter, in the above two scripts. What do you think it is? Try it and see.

We can use the following syntax:

```
$i = ++$i
```

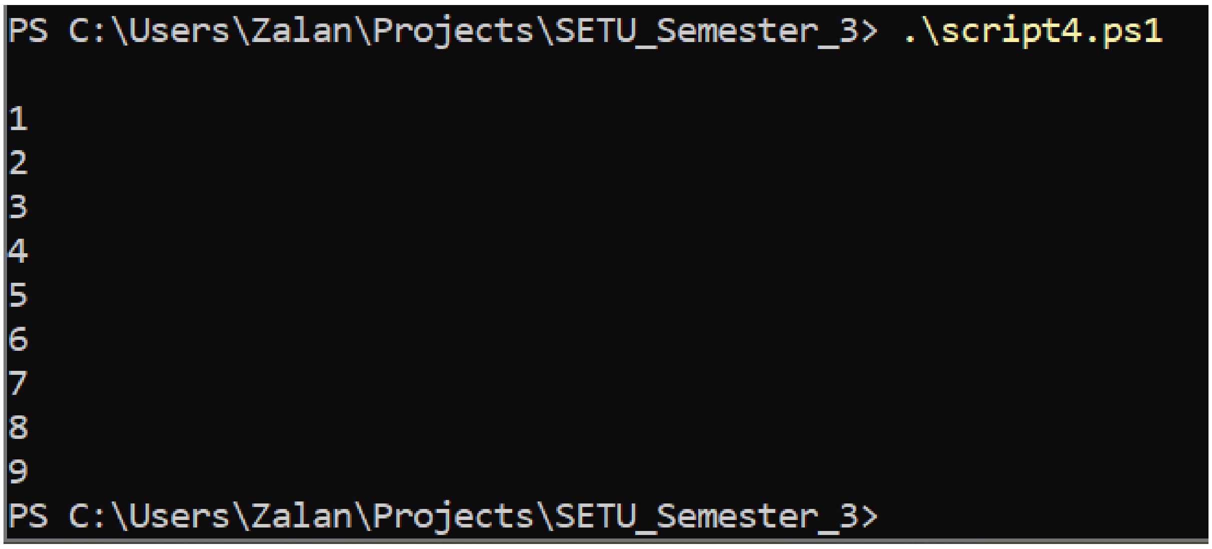
Or here is another version (increasing after):

```
$i = $i++
```

22. Oops, I've put semicolons at the end of each of the lines, in the do...until loop script. I suppose it's because of my experience in writing programs in other languages that sometimes I put semicolons at the end of a line of PowerShell script, even when they're entirely unnecessary in PowerShell. Does PowerShell forgive me for doing this? Find out, and write your conclusion.

```
do
{
Write-Host $i
$i = $i + 1;
}until($i -eq 10)
```

It seems to not bother Powershell at all. It tends to ignore it and run it just as it should.



```
PS C:\Users\Zalan\Projects\SETU_Semester_3> .\script4.ps1

1
2
3
4
5
6
7
8
9
PS C:\Users\Zalan\Projects\SETU_Semester_3>
```

23. Alter this program to deal with grade categories (for example >70 is a distinction mark etc) in an examination and also allow the user to enter a grade.

Original:

```
$temperature = 1
switch($temperature)
{
 { $_ -lt 0 } { "Below Freezing"; break }
 { $_ -eq 0 } { "Exactly Freezing"; break }
 { $_ -le 10 } { "Cold"; break }
 { $_ -le 20 } { "Warm"; break }
 default { "Hot" }
}
```

Rewritten:

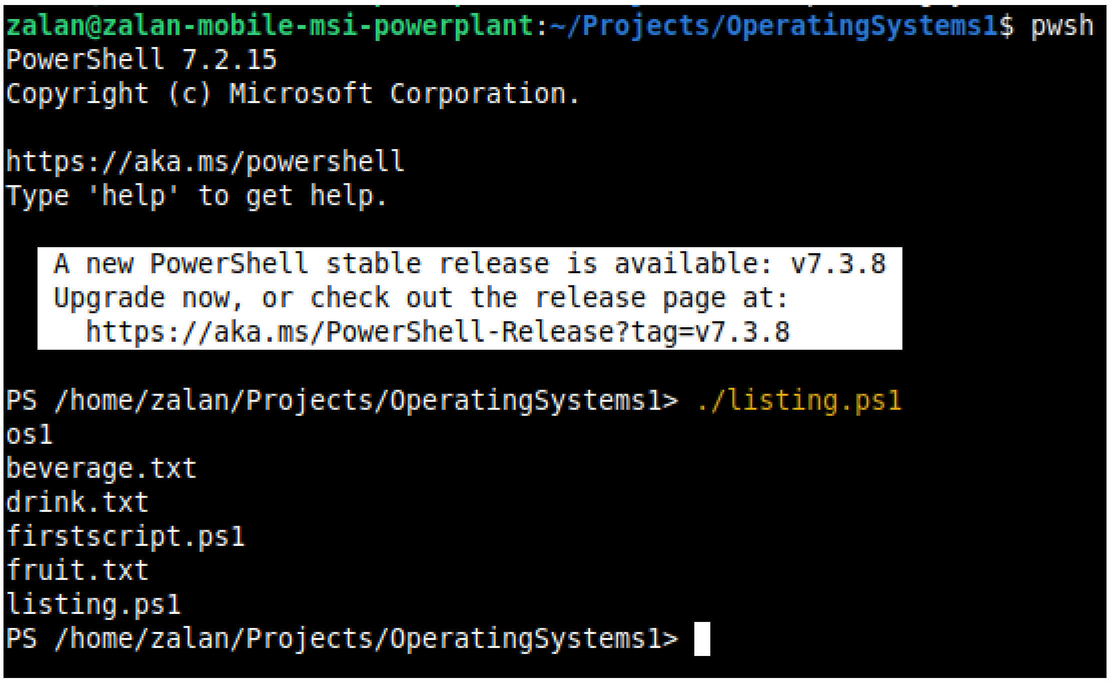
```
do
{
 write-host -nonewline "Enter a percentage between 0 and a 100
inclusive: "
 $inputString = read-host
 $value = $inputString -as [Int]
 $between = $value
 $ok = ($between -ge 0) -and ($between -le 100)
 if ( -not $ok ) { write-host "You must enter number between 0 and 100
inclusive" }
}
until ( $ok )
write-host "You entered: $value%"

$mark = $value
switch($mark)
{
 { $_ -lt 40 } { "Fail"; break }
 { $_ -lt 50 } { "Third Class"; break }
 { $_ -lt 60 } { "Lower Second Class"; break }
 { $_ -lt 70 } { "Upper Second Class"; break }
 default { "First Class" }
}
```

24. Explain what the example below does. Modify it to show fields other than the Name field.

```
$listing = dir
$howLong = $listing.Length
```

```
$i = 0;
while($i -lt $howLong)
{
Write-Host $listing[$i].Name
$i++
}
```

It lists the contents of the current directory. It stores those items in an array ($listing) and stores the amount of items in $howLong. Then it iterates through each item, outputting each element to the terminal.



```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$ pwsh
PowerShell 7.2.15
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

   A new PowerShell stable release is available: v7.3.8
   Upgrade now, or check out the release page at:
     https://aka.ms/PowerShell-Release?tag=v7.3.8

PS /home/zalan/Projects/OperatingSystems1> ./listing.ps1
os1
beverage.txt
drink.txt
firstscript.ps1
fruit.txt
listing.ps1
PS /home/zalan/Projects/OperatingSystems1>
```

25. Draw up a chart to show equivalent syntaxes for different control structures/data structures among Java, Unix script, and PowerShell. You'll have to revisit this question when you've learned some Unix/Linux.

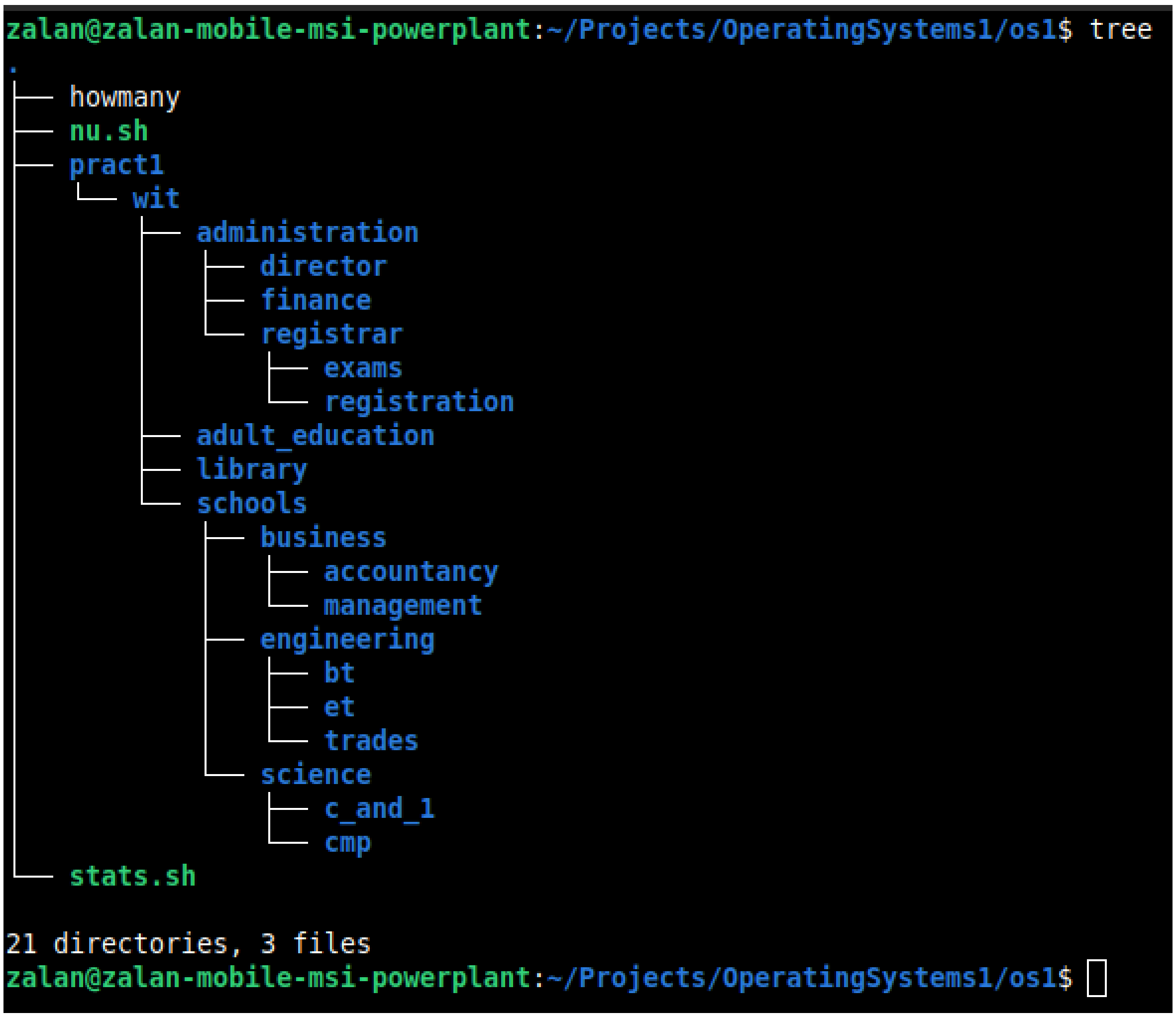
Structure	Java	Bash	PowerShell
If statement	if (condition) { ACTIONS }	if [condition]; then ACTIONS fi	if (condition) { ACTIONS }
Else statement	else { ACTIONS }	else ACTIONS fi	else { ACTIONS }

For loop	for (init; condition; update) { ACTIONS }	for variable in values; do ACTIONS done	for ($init; $condition; $update) { ACTIONS }
While loop	while (condition) { ACTIONS }	while [condition]; do ACTIONS done	while (condition) { ACTIONS }
Array	int[] array = new int[size]	array=("value1" "value2")	$array = @("value1", "value2")
Accessing an array	array[index]	${array[index]}	$array[index]

26. Compare the ways in which scripts are enabled to run in (a) Unix/Linux and (b) PowerShell. Again this is a question for review when you've done some Unix/Linux.

a) When we want to run a script in Linux, we have to make it executable first. It can be set with the chmod command. In order to run the file, we can refer to the file's location (./mypath/myscript.sh).
b) PowerShell uses an execution policy in order to protect its users from malicious scripts. This policy can be set by the Set-ExecutionPolicy command. We can run the command in PowerShell by specifying the file (.\myscript.ps1) or we can call the script directly with the command: powershell -File script.ps1.

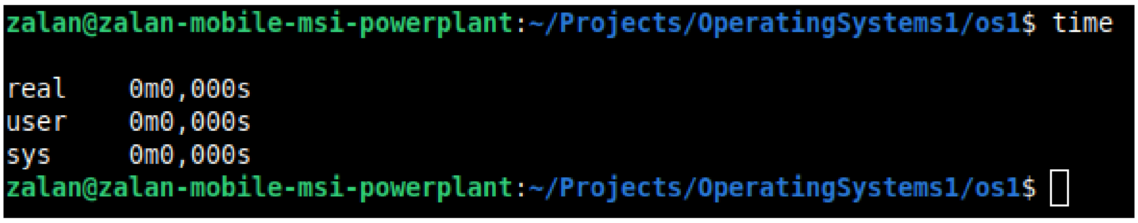
27. Create a directory tree structure (Similar to the structure outlined in Practical 1.



```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ tree
.
├── howmany
├── nu.sh
├── pract1
│   └── wit
│       ├── administration
│       │   ├── director
│       │   ├── finance
│       │   └── registrar
│       │       ├── exams
│       │       └── registration
│       ├── adult_education
│       ├── library
│       └── schools
│           ├── business
│           │   ├── accountancy
│           │   └── management
│           ├── engineering
│           │   ├── bt
│           │   ├── et
│           │   └── trades
│           └── science
│               ├── c_and_l
│               └── cmp
└── stats.sh

21 directories, 3 files
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$
```

28. Find out what happens when you type time at the command line in a UNIX and Windows CLI

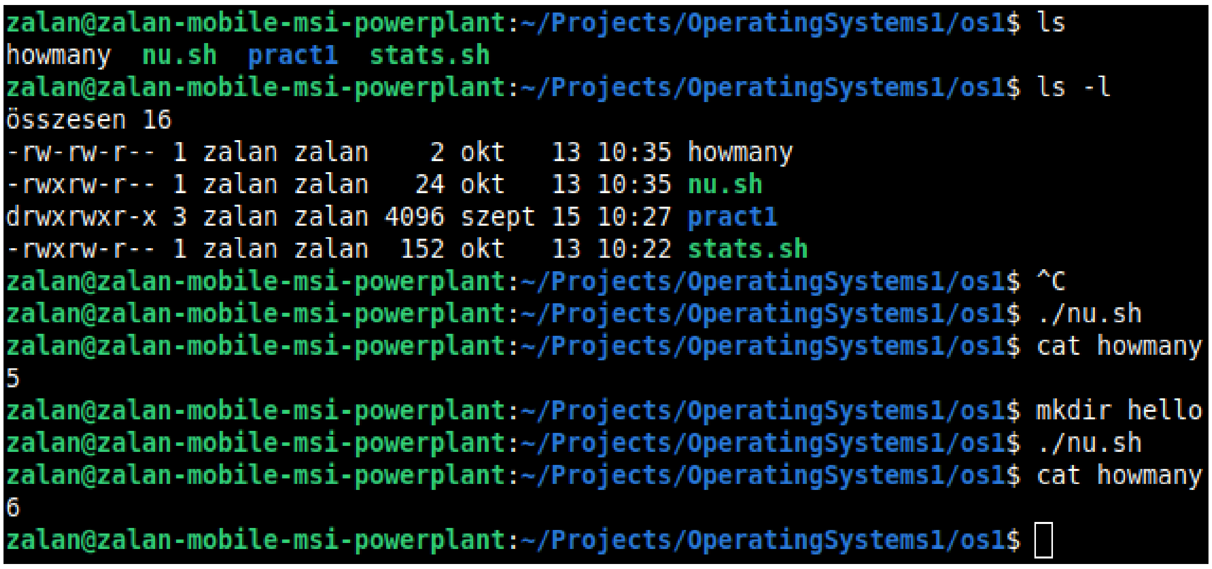
In Unix, the time command provides the following:
It is used to run programs and summarize system resource usage.
It determines which information to display about the resources used by the COMMAND from the string FORMAT. If no format is specified on the command line, but the TIME environment variable is set, its value is used as the format. Otherwise, a default format built into time is used. In Windows, it just shows the current time.



```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ time

real	0m0,000s
user	0m0,000s
sys	0m0,000s
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$
```

29. As an extension to the script NU.SH you used the shell script in conjunction with the file redirection operator (>) to redirect the output to a file called howmany. Run the command again but this time with the >> operator. Check the contents of the file and explain the output

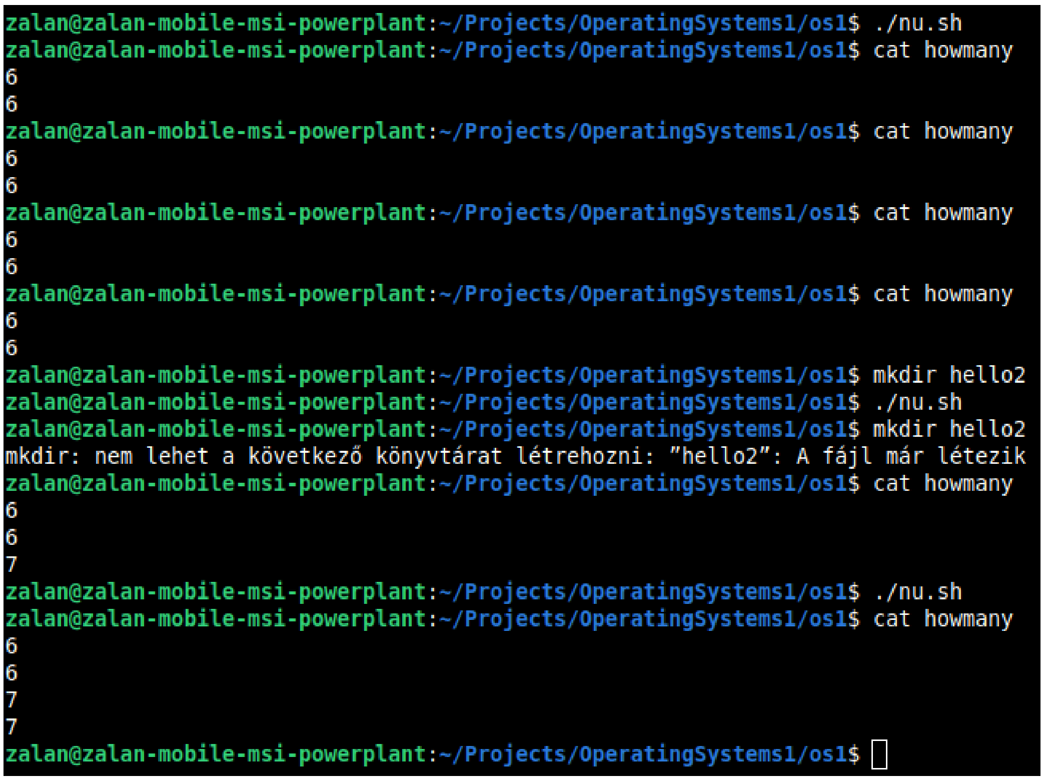
```
ls -l | wc -l > howmany
```



```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ls
howmany  nu.sh  pract1  stats.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ls -l
összesen 16
-rw-rw-r-- 1 zalan zalan    2 okt   13 10:35 howmany
-rwxrw-r-- 1 zalan zalan   24 okt   13 10:35 nu.sh
drwxrwxr-x 3 zalan zalan 4096 szept 15 10:27 pract1
-rwxrw-r-- 1 zalan zalan  152 okt   13 10:22 stats.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ^C
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ./nu.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
5
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ mkdir hello
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ./nu.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
6
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$
```

If we use this with the >> operator, it'll write the output to a new line.

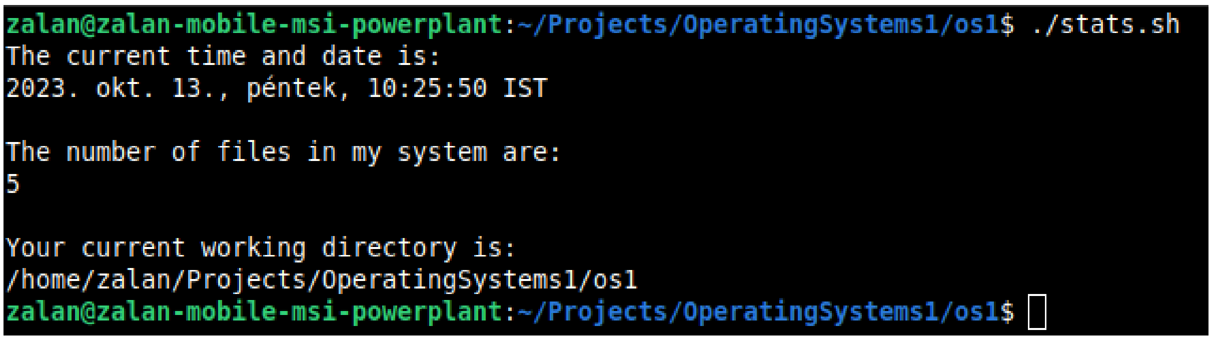
```
ls -l | wc -l >> howmany
```



```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ./nu.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
6
6
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
6
6
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
6
6
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
6
6
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ mkdir hello2
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ./nu.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ mkdir hello2
mkdir: nem lehet a következő könyvtárat létrehozni: "hello2": A fájl már létezik
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
6
6
7
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ./nu.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ cat howmany
6
6
7
7
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$
```

30. Show the finished shell script code (including comments) for the Example 2 exercise. To show date and time, the number of files and directories, and your current working directory.

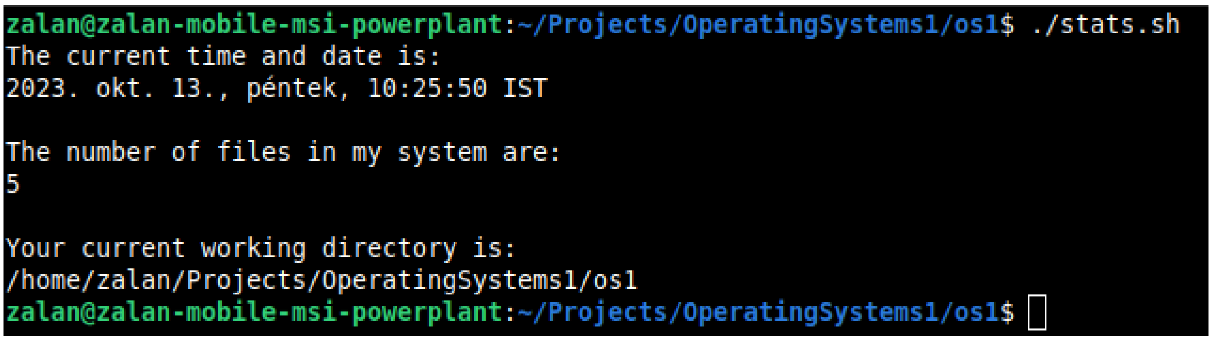
```
# Shows the date and time
echo The current time and date is:
date
echo
# Shows the number of files in the directory
echo The number of files in my system are:
ls -l | wc -l
echo
# Shows the current working directory
echo Your current working directory is:
pwd
```



```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ./stats.sh
The current time and date is:
2023. okt. 13., péntek, 10:25:50 IST

The number of files in my system are:
5

Your current working directory is:
/home/zalan/Projects/OperatingSystems1/os1
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$
```

31. What's the output of the STATS.SH program, and what's the significance of the echo statement on a line without any succeeding text?

The echo statement outputs in new lines, so the output of the bash file will be more readable.



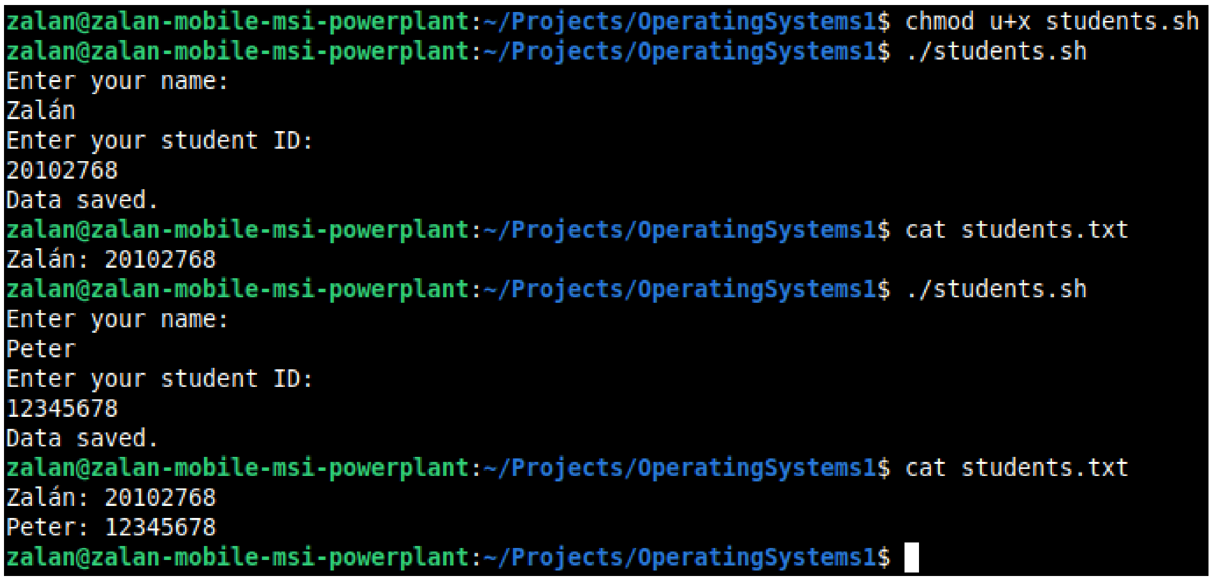
```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$ ./stats.sh
The current time and date is:
2023. okt. 13., péntek, 10:25:50 IST

The number of files in my system are:
5

Your current working directory is:
/home/zalan/Projects/OperatingSystems1/os1
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/os1$
```

32. Write a shell script (STUDENTS.SH) to ask the User to enter two fields. Their NAME and STUDENT-ID. The script should append the data to a text file called STUDENTS.TXT.

The written shell script:

```
echo "Enter your name:"
read name

echo "Enter your student ID:"
read student_id

echo "$name: $student_id" >> students.txt
echo "Data saved."
```

Running the script:



```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$ chmod u+x students.sh
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$ ./students.sh
Enter your name:
Zalán
Enter your student ID:
20102768
Data saved.
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$ cat students.txt
Zalán: 20102768
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$ ./students.sh
Enter your name:
Peter
Enter your student ID:
12345678
Data saved.
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$ cat students.txt
Zalán: 20102768
Peter: 12345678
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$
```

33. Find out the output:

a) ls ~

Lists the contents of my user directory.

b) ls ~/..

Lists the contents of the parent directory of my user directory.

c) cd ~/

Changing the current working directory to my user directory.

34. Using the provided method create another file called list 2 containing the following fruit: orange, plum, mango, grapefruit. List the contents of list 2.:

```
zalan@zalan-mobile-msi-powerplant:~$ cat > list2
orange
plum
mango
grapefruit
zalan@zalan-mobile-msi-powerplant:~$ cat list2
orange
plum
mango
grapefruit
```

35. Find out what happens if you do:

```
cat >> mondaylist
rabbit
hare
^d
```

a) When the file doesn't exist

Creates the mondalist file and appends rabbit and hare to the file.
File contents:

```
rabbit
hare
```

b) When the file already exists

Appends rabbit and hare to the mondaylist file.
File contents:

```
rabbit
hare
rabbit
hare
```

36. Using pipes, display all lines of list1 and list2 containing the letter 'p' and sort the result.

```
$grep 'p' list1 list2 | sort
```

OUTPUT:

```
list1:apple
list1:eper
list1:pear
```

This output is correct as only list1 in my example contained the letter p.

37. Using pipes and filters search all files ending in *.txt in your linuxstuff directory that contain the names of ALL other users in your class currently logged in. Output the results in ascending order to Standard Output.

grep 'zalan' ./linuxstuff/*.txt | sort

```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1$ grep 'zalan' ./linuxstuff/*.txt | sort
./linuxstuff/data2.txt:zalan
./linuxstuff/data.txt:zalan
```

38. Explain what the following do:

a) ls linuxstuff/backups

Lists the contents of the backups directory located in the linuxstuff directory in the current working directory

b) ls ~/linuxstuff

Lists the contents of the linuxstuff directory in the user's home directory.

c) ls ~

Lists the contents of the user's home directory.

d) ls ~/..

Lists the contents of the parent directory of the user's home directory.

e) cat > list1 … ^D

Creates or overwrites the file list1 with the input provided by the user until ^D (Ctrl+D) is pressed

f) cat >> list1 … ^D

Appends the input provided by the user to the end of the file list1 until ^D (Ctrl+D) is pressed.

g) cat list1

Displays the contents of the file list1

h) cat list1 list2 biglist

Adds the 3 file contents together and displays it.

i) sort … ^D

Sorts the input provided by the user until ^D (Ctrl+D) is pressed and outputs the sorted result.

j) sort < biglist

Sorts the contents of the file biglist and displays the sorted output.

k) sort < biglist > slist

Sorts the contents of the file biglist and writes the sorted output to the file slist.

l) ls list*

Lists all files in the current directory that start with list.

m) ls *list

Lists all files in the current directory that ends with list.

n) ls ?list

Lists all files in the current directory that have exactly one character before list.

o) who

Displays information about users currently logged into the system.

p) who > names.txt

Redirects the output of the who command to the file names.txt, creating or overwriting it.

q) sort < names.txt

Sorts the contents of the file names.txt and displays the sorted output.

r) who | sort

Sorts the output of the who command.

s) who | wc –l

Counts the number of lines in the output of the who command, counting the number of users currently logged in.

t) grep 'orange' list2

Searches for the word orange in the file list2 and displays the lines where it is found.

u) grep '^grape' list2

Searches for lines in the file list2 that start with grape.

v) grep 'grape$' list2

Searches for lines in the file list2 that end with grape.

39. Write a note to explain the difference between ~ and / in describing directories.:

~ stands for the current user's home directory
/ stands as a directory separator in file paths or stands for the root directory, depending on the context.

Practical 6

Shell Programming Assignment

The code for the scripts is attached separately in the ZIP file.
The extra login attempt checker has been added.

Example storage for registration data (Register.txt):

```
<USR-zalan><PWD-nem><Zalán><01/01/2000><zalant@hello.net>
<USR-hello><PWD-nono><Hello><22/02/1998><something@smt.com>
```

Example usage of the scripts:

```
zalan@zalan-mobile-msi-powerplant:~/Projects/OperatingSystems1/linuxproject$
./Menu.sh

Welcome, please choose one of the following options:
1. Register your details
2. Login to your account
3. Exit
Enter your choice: 1
Enter your full name: Zalán
Enter your date of birth (DD/MM/YYYY): 01/01/2000
Enter your username: zalan
Enter your password:
Enter your email address: zalant@hello.net
Registration successful.

Welcome, please choose one of the following options:
1. Register your details
2. Login to your account
3. Exit
Enter your choice: 2
Enter your username: zalan
Enter your password:
Attempting login.
Logged in!

You are currently logged in. Choose option from below:
1. Logout
Enter your choice: 1
Logging out.

Welcome, please choose one of the following options:
1. Register your details
```

```
2. Login to your account
3. Exit
Enter your choice: 2
Enter your username: gwwff
Enter your password:
Login failed.
Enter your username: dw
Enter your password:
Login failed.
Enter your username:
Enter your password:
Login failed.
Maximum login attempts reached.

Welcome, please choose one of the following options:
1. Register your details
2. Login to your account
3. Exit
Enter your choice: 1
Enter your full name: Hello
Enter your date of birth (DD/MM/YYYY): 22/02/1998
Enter your username: hello
Enter your password:
Enter your email address: something@smt.com
Registration successful.

Welcome, please choose one of the following options:
1. Register your details
2. Login to your account
3. Exit
Enter your choice: 2
Enter your username: hello
Enter your password:
Attempting login.
Logged in!


You are currently logged in. Choose option from below:
1. Logout
Enter your choice: 1
Logging out.

Welcome, please choose one of the following options:
1. Register your details
2. Login to your account
3. Exit
Enter your choice: 3
Exiting program.
```

Practical 7

Things to try part 1.1

Threading Program 1.1

Create an Eclipse Java Project to use this program and run the software 10 times, each time copying and pasting your results into a word processor file. Label this set of 10 results clearly with the heading *Threading Program 1.1*

Do you see much variation on your results from one run to the next? Explain what conclusions you come to.

Outputs in order:

1. aaaaaaaaaaaaaaaaaaaabbbbbbbbbaaaaaaaaaaaaaaaaaabbaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaabbbbaaaaaaaaaaaaaaaaaaabbbbbaaabbbbbbbbbbbbb bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
2. aaaaaaaaaaaaaaaaaabbbbbbbbbbbbabbaaaaaaaaabbbbbbbbbbbbbbbaaaaaaaaab bbbbbbbbaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbaaaaaaaaabbbbbbbaaaabbbb bbbbaaaaaaaaaabbbbbbaaaaaaaaaabbbbbbbbbbbaaaaaaaaabbbbbbbbb 1 2 3 4 5 6 7 8 9aaaaaabbb 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
3. aaaaaaaaaaabbbbbbbbbaaaaaaaaaaaaabaaaaaaaaaaabbbbbbbbbaaaaaaaaaabbbb bbbbbaaaaaaaaaaaaabbbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaabbbbbbbaaaaaabb bbbbbbaaaaaaaaaabbbbbbaaaaaaaaaaaa 1bbbbbbbbbbbb 2bbbbbbbb 3bbbbbb 4bbbbbb 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
4. aaaaaaaaaaaaaaaaaaaaaaabbbbbbbbaaaaaaaabbbbbbaaaaaaaabbbbbbbbbbaaaaa aabbbbbbbbbbbbbbbaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaabbbbbbbbbaaa aaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbaaaaaaaabbaaaaaaabbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
5. aaaaaaaabbbbbbbbbaaaaabbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaabbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbb bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

6. aaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaabbbbabbbbbbbbbbbbaaaaaaaabb bbbaaaaaabbbbbbbbbbbbbbaaaabbbbbbbbaaaaaaaaaaaaabbbbbaaaaaaaaabbbbb bbbaaaaaabbbbbbbbaaaaaaabbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaabbb 1aa 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
7. aaaaaaaaaaaabbbbbbbbbabbaaaaaaaabbbbbaaaaaaaabbbbbbbbaaaaaaaaaabbbb bbbaaaaaaaaaabbbbbbbbbbaaaaaaaabbbbbbbbaaaaaaaabbbbbbaabbbbbbbbbaa aaaaaabbbbbbbbaaaaaaaaaaaabbbbaaaaaaaaaaaabbbbbbabbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
8. aaaaaaaaaaabbbbbbbbbbaaaaaaaaaaabbaaaabbbbbbbbbbbbbbaaabbbbbbbbbbaa aaaaabbaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaabbbbbbbbbbbbb bbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1aaa 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
9. aaaaaaabbbbbbbbaaaaaabbbaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbb bbbbaaaaabaaaaaaaaaaaaaaaaaaaaaaaaabbbbbaaaaaaaaabbbaaaaaaaaabbbbaaaaa aaaaabbbbbbbbbbbbaaaaaaaaaaaaabbbbbb 1 2bbbb 3 4bbbbbbbbbbbbb 5bbbbbbbbb 6bbbbbb 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
10. aaaaaaaaaaaaaaaaabbbbbbbbbaaaaaaaaaaaabbaaaaaaaabbbbbbbbbaaaaaaaaaa bbbbbbbaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbbbbbaaaaaaaaabbbbbbbbbaaa aaaabaaaaaaaaaabbbbbbbbbbbbbbaaaaaaaaaabbbbbbbaaaabbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

The outputs differ as the 3 functions run on different threads, which are asynchronous to each other. That means there's no guarantee for consecutive run, which we can see here.

Running functions single-threaded

Replace the lines:

```
thread1.start();

thread2.start();

thread3.start();
```

with

```
thread1.run();

thread2.run();

thread3.run();
```

Does your software still run?

Yes, the program still, run, but in that case in single-threaded way.

If it does, what difference do you see?

The output is now consistent (functions run after each other, syncronously), which means that the functions run in single-threaded way.

The single-threaded output every single runtime:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbb bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Things to try part 1.2

Threading Program 1.2

Create an Java Eclipse Project to use this program and run the software 10 times. Each time copy and paste your results into a word processor file. Label this set of 10 results clearly with the heading *Threading Program 1.2*

Outputs in order:

1. aaaaaaabbbbbbbbbbaaaaaaaaacccccccccbbaaaaaaaaacccccccccbbbbbbbbbbaaaa cccccccbbbbbbbbbaaaaaaaaaacccccccccbbbbbbbbbaaaaaaaaacccbbbbbbbbaaaaa aaaaacccbbaaaaaaaaaabbbbbbbbbbbbbbbaaaaa 1bbbbbbaaaaaaaa 2bbbbbaaaaaaaaaaa 3bbbbbbaaaaaaaa 4bbbbbbb 5 6bbbbbbbb 7b 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

2. aaaaaaaaabbbbbbbbcccccccccccaaaaaaaaaabbbbbbbbbccccccccaaaaaaaab bbbbbbbbcccccccaaaaabbbbbbbbbcccccccaaaaaaaabbbbcccccccaaabbbbbbbbaaaa aaaabbbbbbbbbaaaaaaaaabbbbbbbbaaaaaabbbbaaaaaaaaabbbbbbbbaaaaaabbbbbb bbaaaaaaaaabbbbbbbbaaaaaaaabbbbbbaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
3. aaaaaaaaccbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaacccccccccccbbbb bbbbbbbbbbaaaaaaaacccccccccbbbbbbbbbaaaaaaaaacccbbbbbbbbbaaaaaaaaccc ccccbbbbbbbbbaaaaaaaaaaaacccccccbbaaaaaaaabbbbbbbbbaaaaaaaaa 1bbbaaaaaaaa 2bbbbbbbbbaaaaaaaa 3bbbbbbbbbba 4a 5 6 7 8 9aa 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
4. aaaaaaabbbbbbbbccccccccccaaaaaaaabbbcaaaaaaaabbbbbcaaaaaaaaabbbbbbbb ccaaaaaaaabbbbbbbbcccccaaaaaaabbbbbbbcccccaaaaaaaaaaaaaaaaaaaaaaa aaaabcaaaaaaabbbbbbbbbbbbbcccccaaaaaaaaaaabbbbbbbbbbbbcccccccaaaa aabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
5. aaaaaaaaabbbbbbbbbbbbbbbbbbbcccaaaaaaaaabccccccccccccccccaaaaaaabbbbb bbbbbbbbbbcccccccccccccccccccccaaaaaabaaaaaaaaaaaaabbbbbbaaaaaabbbbbbb aaaaaaaabbbbbbbaaaaaaaaabbbbbbbabbbbbbbaaaaaaaabbbbbbbbaaaaaaaaaabbbb bbbbbbbaaaaaaaaaabbbbbbaaaaaaabbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
6. aaaaaaaaaabbbbbbbbbbcccccccaaaabbccccccaaaaaabbbbbbbbbcccccccccaaaaaaaa abbbbbbbbcccccccccaaaaaaaaabbbbbbbbbcccccccaaaaaaaaaabbbcccaaaaaaaaa aaaaaaaaaabbbbbbbaaaaaaaaaaaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbaaaab bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
7. aaaaaaacccccccccbbbbbbbbaaaaaaaacccccccccbbbbaaaaaaaaaacccccccccbbbbbbbb baaaaaacccccccccccbbbbbbbbbbaaaaaaaacccbbbbbbbbbaaaaaaaabbaaaaaabbbbbbb bbaaaaaaaaaabbbbbbbaaaaaaaaabbbbbbbbbbbaaaaaaaaaaabbbbbbbbbbbbaaaaaab bbbbbbbbbbbbbbbbb 1 2aaaaaaaa 3 4 5aaaaaaaa 6a 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

8. bbbbbbbbbaaaaaaaaaaaaaaaacbbbbbbbbbaaaaaaaaccccbbbbbbbbaaaaaaaccc cccccccccccccccccccccccccccccccbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb bbbbbbbbaaaaaaaaabbbbbbaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
9. acccccccccbbbbbbbbbaaaaaaaaaaaaaaaacccccccccccbbbaaaaaaaaaaaacccccccbb bbbbbaaaaaaacccbbbbbbbbbbaaaaaaaabbbbbbbbccccccccccaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaabbbbbaaaaaabbbbbbbbbbbbbbbbbbbbbaaaaaaaabbbbbbbbbbaaa aaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
10. aaaaaaaaabbbbbbcccccccccaaaaaaaaaaaaaaaaaaaaaaabbbbbbcccccccccaaaaaaab bbbbbbbccccccccccaaaaaaabbbbbbbbcccccccaaaaaaaaabbbbbbbbbccccccaaaaaaa aabbbbbbbbbaaaaaaaaaabbbbbbbbbbbbaaaaaaaaaabbbbbbbbaaaaaaaaaaaaabbbbb bbaaaaabbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Do you see much variation in your results from one run to the next? Explain what conclusions you come to.

The outputs differ as the 4 functions run on different threads, which are asynchronous to each other.
Probably I should've got a bit different result, but it's likely that the CPU handled the process too fast.

Things to try part 1.3

Threading Program 1.3

Create an Eclipse Java Project to use this program and run the software 10 times, each time copying and pasting your results into a word processor

Outputs in order:

1. aaaaaaaaaabbbbbbbbbbbbaaaaaaaaaaaaabbbbbbbbbbabbbbbbbbbbbaaaaaabbbbbb baaaaaaaabbbbbbbaaaaaaaaaaabbbbbbbaaaaaaaaaabbbbbbbaaaaaaaaabbbbbbbaaa bbbbbbbbaaaaaaaaabbbbbbbbaaaaaaaabbbbbbbbaaaaaaaaabbbbbbbbbaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

2. aaaaaaaabbbbbbbbaaaaaaaaaaabbabaaaaabbbbbaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaabbbbbbbbbbbbaaaaaaaabbbbbbaaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
3. aaaaaaaaaaaaabbbbbbbaabbbbbaaaaaaaabbbbbbbbbbbbbbbbbbbaaaaaaabbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbbbbbbbbaaaabbbbbbbaaaaabbbbbbbaaaabbbbbbbaaaaaaaaaabbbbbbaaaaaaaaaaabaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
4. aaaaaaaaaaabbbbbbbbbbaaaaaaaaaaaaaaabbaaaaaaaaaaaaabbbbbabbbbbbaaaaaaaaaaaaaaaaaabbbbbbbbbaaaaaaaaaaaaabbbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaabbbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
5. aaaaaaaaaabbbbbbbbbbbbbbaaaaaaaaaabbbbbbbbbbbaaaaaaaaaaaaaabbbbbbbbbaaaaaaaaabaaaaaaaaaabbbbbbbbaaaaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaabbbbbbbaaaaabbbbbbbbaaaaaaaaaaaaaaabbbbbbbaaabbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
6. aaaaaaaaaabbbbbbbbbbbaaaaaaaaabaaaaaaaabbbbbbbbbaaabbbbbbbaaaaaaaaaabbbbbbbbbaabbbbbbbbbbaaaaaaaaaabbbbbbbbaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbb 1aaaaaaaabbbbbbbbbbb 2 3 4aaaaaaaaaaabbbbbbb 5 6aaaaaaaaaabbbbb 7aaaaabbbbbbb 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
7. aaaaaaaaaabbbbbbbbbbbaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaabaaaaaaabbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1aaaaaaaaaaaa 2aaaaaa 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
8. aaaaaaaaabbbbbbbbaaaaabaaaaaaaaaaabaaaaaaabaaaaaaaaaaaabaaaaaaaaaabbbbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaaaabbbbbbbbaaaaaaaaabbbbbbbbbbaaaaaaabbbbbbbbbaaaaaaabbbbbbbbbbbbbbaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbb 1

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

9. bbbbbbaaaaaaaaabbaaaaaaaaaabbbbbbaaaaaaaaabbaaaaaaabbbbbbbbbaaaa aabbbbbbbbbbaaaaaaaaabbbbbbbbbbaaaaaaaabbbbbbaaaaaabbbbbbbbbaaaaa aaaabbbbbbaaaaaaaabbbbbbaaaaaaaaaaaabbbbbbbbaaaaaaabbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
10. aaaaaaabbbbbbbbbaaaaaabbbbbbbbbaaaaaaaaaaaabbbaaabbbbbbbbbbbaaaaaaaa abbbbbbbaaaaaaaabbbbbbbbbbbbaaaaaaaabbbbbbaaaaaaaabbbbbaaaaabbbbb bbbaaaaaaaaaaaaaaaaabbbbbbbbbbaaaaaaaabbbbbbbaaaaaaaabbbbbbbbaabb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Do you see much variation in your results from one run to the next? Explain what conclusions you come to.

The outputs differ as the 3 functions run on different threads, which are asynchronous to each other.

Things to try part 1.4

Threading Program 1.4

Create a JavaEclipse Project to use this program and run the software 10 times, each time copying and pasting your results into a word processor file. Label this set of 10 results clearly with the heading *Threading Program 1.4*

Outputs in order:

1. aaaaaaaaabaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaabbbbbbbbbbaaaa aaaaaaaaaaaaaaaaaabbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaabbbbbbbbaaaaaaaabb aaaaaaaaaaaaaaa 1bbbbbbbbaaaaaaaaa 2 3bbbbbbbb 4bbbbbbb 5bbbbbbb 6bbbbbbbbbbb 7b 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
2. aaaaaaabbbbaaaaaaaaaabbbbbbbbbaaaaabaaaaaaaabaaaaaaaaaaaaaaaaaaaabbb bbbbaaaaaaaaaaaaabbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaabbbbbbbbbbbbbbaaa aaaaaaaaaaaabbbbbbbbbaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

3. aaaaaaaaaabbbbbbbbbaaaaaaaaaabbaaaaaaabbbbbbbaabbbbbbbbbbbbbaaaaa aabbbbbbaaaabbbbaaaaabbbbbbbbbbbbbbaaaaaaaabbaaaaaaabbbbbbbbbbbba aabbbbbbbbbaaaaaaabbbbbbbbbaaaaaaabbbbbbbaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
4. abbbbbbbbbaaaaaabbbbbbbbbbaaaaaaaaaaabbbbbbbbbaaaaaaaaaabbbbbbbba aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbaaabbbbbbbbaaaaaaaaaaab bbbbbbbbbbbbbbaaaaaaaaabbbbbbbbbbbaaaaaaaaabbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
5. aaaaaaabbbbbbbbbaaaaaabaaaaaaaaaaaabbbbbbbbbbbbbbbaaaabbbbbaaaaaaa bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaabbbbbaaaaaabb baaaaaaabbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
6. aaaaaaaabbbbbbbbbaaaaabbaaaaaaaaaaabbbbbbbbbbbbbaaaaaaaaaabbbbbbbaa aaaaaabbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaabbbbbbbbbaaaaaabbbbbbaaaaaabb bbbbbbbbbbaabbbbbbbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaabaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
7. aaaaaaaaabbbbbbaaaaaaaaaaaaaaaaaaaabbbbbbaaaaaaaaaaaaaaaabbbbbbbaaaaa aaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbaaaaaabbbbbbbaaaaaaabbbbbbbbaaa aaaabbbbbbbbbaaaaaaabbbbbbbbbbaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
8. aaaaaaaaaabbaaaaaaaaaaaaaaaabbbbbbbaaaaaaaaabaaaaaaaaaabaaaaaaaaaabbbb aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbb bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
9. aaaaaaaaabbbbbbaaaaaabbbbbbaaaaaaaaaabbbaaaabaaaaabbbbbbbbbbbbabbbaaa aaaabbbbbbbbbaaaaaaaaaaabbbbbbbbaaaaaaaabbbbbbaaaaaaaaaabbbbbbbbaaaa aaaaabbbbbbbaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbbbaaabbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58

59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

10. aaaaaaaaaabbbbbbbbbbaaaaaaaaabbaaaaaaaaaaaabbbbbbbbbbaaaaaaaaaaabaaaaa aaaabbbbbbbaaaaaaaaaaabbbbbbbaaaaaaaaaaaaaabbbbbbbbaaaaaaaaaabbbbbbbba aaaaaaaaaaabbbbbaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Do you see much variation in your results from one run to the next? Explain what conclusions you come to.

The outputs differ as the 3 functions run on different threads, which are asynchronous to each other. Even though from the number 50, the number printing slows down, it doesn't make much difference, as the CPU was fast enough to print out the characters first.

Repeat part one above several times, each time with a different sleep time. Write out your conclusions.

The result will be very similar as above, the difference is the printout of the last 50 numbers, which will be slower, depending on the setting. For example 500 milliseconds will make 1 printout of those numbers every half second.

Things to try part 1.5

Threading Program 1.5

Consider now the program for part 1.1 above and, in particular, the following lines:

```
// Third code block: Start threads

thread1.start();

thread2.start();

thread3.start();
```

Now change these lines to read:

```
// Third code block: Start threads

thread1.start();

thread2.start();

thread3.setPriority(Thread.MAX_PRIORITY);
```

```
thread3.start();
```

Do the first of the *things to do* of part 1.1 again but using the modified program this time.

Outputs in order:

1. aaaaaaaaaabbbbbbbbbbbbaaaaaaaabbbbbbbbbbbbbabbbbbbbbbaaaaaaaaabbbb aaaaaaaaabbbbbbbbbaaaaaaabbbbbbbbbaaaaaaaaabbbbbbbbbaaaaaaaaaabbbbb bbbbaaaabbaaaaaaaabbbbbbbbbbaaaaaaaaaaaaaaaaa 1bbbbbbaaaaaaaa 2bbbbbbb 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
2. aaaaaaabbbbabbbbbbbbbbbbbbbabbbbbbbbbbbabbbbbbbbaaaaaaaaaaaaaaaaaaaab bbbbbbaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbaaaaaaaaaaabbbbbbbbbaaaaaaabbbb bbbbbbbbbbbbbbaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
3. aaaaaaaaaabbbbbbbbbbbbbaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaabbbbb bbbbbbbbabbbbbaaaaaaaaaabbbbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaaabbbbbbbb baaaaaaaaaabbbbbbbbaabbbbbbbbbbbbb 1aaaaaaaaab 2aaaaaaaaa 3aaaaaaaa 4aaaaaaa 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
4. aaaaaaaabbbbbbbbbaaaaabbaaaaaaaaaaaaabbbbbaaaaaaaaabbbbbbbbbbbbbbbbb bbbaaaaaaabababbbbbaaaaaaaabbbbbbbbbbbbbaaaaaaaabbbbbbbbbbbbbbbbaaaaaaabb bbbbbbbbbaaaaaaaaaabbbbbbbbbbaaaaaaaaaabbbbbbbaaaaa 1aaaaaaaa 2aaaaaaaa 3 4aa 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
5. aaaaaaaaaaaaaaaaaabaaaaaaaaaaaaaaaabbbbbbbbbbbaaaaaaaaaaaaaabbbbbbbbaaaa aaabbbbbbbbbbbbbbbbbbbbbaaaaaaaaabbbbbbbbbaaaaaaaaaaaaaabbbbbbbbbbaaaaa aaaabbbbbbbbbbaaaaaaaaaabbbbbbbbbbaaaaaaaaaaab 1 2 3bbbbbbbbbb 4bbbbbbbb 5bbbbb 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
6. aaaaaaaaaabbbbbbbbbbbabbbbbbbbbbbbbbaaaaaaaaaabbbbbbbbbaaaaaaaaabbbbbbb aaaaaaaabbbbbbbbbaaaaaaaaaaaabbbbbbbbbaaaaaaaaabbbaaaaaaaaaabbbbbbbbbbaaaa bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaaaaaabbbaaaaaaaaaabaaaaaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

7. bbbbaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbaaaaaaaaa aabbbbbbbbbbaaaaaaaaaaaabbbbbbbbaaaaaaabbbbbbbbb 1aaaaaaaabbbbbbb 2aaaaaaaaabbbbbbb 3aaaaaaaaaaaaaabb 4aaaaaaaaaaaaaaaaaaaaaaaaabbbb 5bbbbbbbbbbb 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
8. aaaaaaabbbbbbbbaaaaaaabbaaaaaaaaaaaabbbbbbbaaaabaaaaaaaaaabbbbbbb baaaaaaaaabbbbbbbbbbaaaaaaaaabbbbbbaaaaaaaaaabbbbbbaaaaaaaabbbbbbaa aaaaaaaabbbbbbabbbbbbaaaabbbbbbbaaaaaaabbbbbbaaabbbbbbbbbbbbbbbbb 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
9. aaaaaaaabbbbbbbbaaaaabbbbbbbbbbbaaaaaaaaaaaabbbbbbbbaaaaaaaaabbbbb bbbbaaaaaaaabbbbbbbbbbbbbaaaaaaaabbbbbbbaaaaaaabbbaaaaabbbbbaaaaaaaa bbbbbbbbbaaaaaabaaaaaaaaabaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbaaaaaaa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
10. aaaaaaabbbbbbbbaaaaaaaaaaaaabbbbaaaaaaaaaaaaabbbbbbbbbbbbbbaaaaaaaaab bbbbbbabbbbbbbbaaaaaaabbbbbaaaaaaaaaaaaaaaaaaaabbbbbbbaaaaaaaaabbbbb bbbbbbbbbaaaaaa 1 2 3bbbbbbbbbbbbaaaaaaaaaaaaaa 4 5 6 7 8 9 10bbbbaaa 11bb 12bbbbbbbb 13bbbbbbb 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Compare your results with those that you got the first time.

There is not much difference (although it was more common to see numbers earlier), even though the priority for Thread 3 was set to MAX. The reason for this is probably the speed/cores of the CPU.

Things to try part 2

When you've run the program write a paragraph that explains what you see in terms of what you've learned in lectures about

1. Buffers

Buffer is a middle storage, which in this example contains the product of the Producer and the data which the Consumer consumes. In this case the buffer's storage is limited, therefore it can only hold a fixed amount of data/values.

2. Multiprocessing/Multithreading

Multithreading helps to process things separately from each other. In this case the Consumer and Producer are running on different Threads, therefore they are running in parallel, separately.

3. The producer-consumer problem.

This is a syncronisation problem. The 2 process, Consumer and Product are using the same resource, in this case buffer. The Producer gives the data into the buffer, and the Consumer takes it out. As the buffer has a capacity it can reach maximum capacity. If it reaches the max value, the Producer stops and waits until there will be free place in the buffer (until Consumer reads out the value). If the Consumer takes out all values from the buffer, it will be empty, therefore the Consumer will have to wait for data to be inserted into the buffer (until the Producer produces data and puts it into the buffer).

Things to try part 3

Experiment with different sleep times

Run the above program, experimenting with different sleep times, than 100ms. Try, for example, 1ms, 20ms, 50ms, 200 ms, 300ms.

The same output happens, if I change the sleep time for both threads. The only change is speed. If I change the speed only for one, it'll just result in more waiting on one side.

Explain it cannot be one threaded

Now that you've run the program, can you see clearly (<u>not</u> from the code but simply from what you observe when running it) why this program, simple though it is, cannot be implemented with only one thread. Explain.

The 2 process are running separately, but in parallel. That is crucial for the program, they are dependent on the other, as one produces and one consumes. If we only have one thread we might wait until the end of the world, as there would be no other thread to make the changes to the buffer on the other end.

Eliminate try-catch blocks

Now eliminate both the *try* block and the *catch* block so that the inside of the *while* block just looks like:

```
        counterWindow.setText("" + count);

count++;
```

```
      if(count > 10)

{

count=1;

}
```

After modification, run the program again and explain what you see.

You cannot eliminate the try-catch blocks, as they would be unhandled exceptions, therefore it would not allow to run the program.

Things to try part 4

Consider the following software:

RunnableJob.java

```
package com.cakes;
public class RunnableJob implements Runnable {
@Override
public void run() {
Thread thread = Thread.currentThread();
System.out.println("RunnableJob is being run by " +
thread.getName() + " (" + thread.getId() + ")");
}
}
```

ThreadExample.java

```
package com.cakes;
public class ThreadExample {
public static void main(String[] args)
throws InterruptedException
{
RunnableJob runnableJob = new RunnableJob();
Thread thread1 = new Thread(runnableJob);
thread1.setName("thread1");
thread1.start();
Thread thread2 = new Thread(runnableJob, "thread2");
thread2.start();
Thread thread3 = new Thread(runnableJob);
thread3.start();
Thread currentThread = Thread.currentThread();
System.out.println("Main thread: " +
currentThread.getName() + "(" +
currentThread.getId() + ")");
}
}
```

Edit the software above and instead of using *`getName()`* and *getID()* use:

```
thread = Thread.currentThread();
System.out.println(thread);
```

The last line will make thread use the Thread class's toString method

Write down what you see.

The output:

```
Main thread: Thread[#1,main,5,main]
RunnableJob is being run by thread1 (30)
RunnableJob is being run by thread2 (31)
RunnableJob is being run by Thread-1 (32)
```

We can see 3 Jobs being run all by separate threads, one is named thread1, other is thread2 and the last one gets a default name Thread-1. The sequence of the output may vary, as they run in parallel, as can be seen below:

Example other output:

```
Main thread: Thread[#1,main,5,main]
RunnableJob is being run by thread2 (31)
RunnableJob is being run by Thread-1 (32)
RunnableJob is being run by thread1 (30)
```