# Question 1 (Utility functions)

function Header

{

param($Header)

Write-Host $Header.ToUpper()

Write-Host "".PadLeft($Header.Length, '=')

} #Header

# Question 2

2.1 function ListFiles

{

param($folder, $format)

# list

if (!$format)

{

$files = get-childitem -Path $folder | Sort-Object BaseName

} else {

$files = get-childitem -Path $folder `

| Where-object {$\_.Extension -eq ("." + $format)} `

| Sort-Object BaseName

}

#Counters

$i = 0

$nFiles = $files.Count

#Looping 10 files per batch do

{

#Header of subset

Clear-Host

Header "List of files"

Write-Host "Folder: " $folder

Write-Host ("Progress: " + $i.ToString() + "/" + $nFiles)

#Subset of files

$files `

| select-object BaseName, Extension, `

@{N = " Size (kb)"; `

E = { ($\_.Length / 1Kb).ToString("n1").PadLeft(10) } `

} `

-Skip $i `

-First 10 `

| Format-Table

#Press Enter ~~for next batch~~ if ($i + 10 -lt $nFiles)

{

$key = Read-Host "Press 'Enter' for next 10 files"

}

$i += 10

} while ($i -le $nFiles)

} #ListAll

2.2 function ListPerFormat

{

param([string] $downloads) $format = Read-Host "Format"

ListFiles $downloads $format

} #ListPerFormat

2.3 function DeletePerFormat

{

param([string] $folder)

Clear-Host

Header "Delete files"

$format = Read-Host "Format"

$files = Get-ChildItem -Path $folder `

| Where-object {$\_.Extension -eq ("." + $format)} foreach ($file in $files ) {

$Confirm = Read-Host `

-Prompt ("Are you sure you want to delete " + $file + " (Y/N)?") if ($Confirm -eq "Y")

{

Remove-Item `

-Path $file.Fullname `

-ErrorAction SilentlyContinue ` -ErrorVariable errors

if ($errors -ne $null) { Write-Host $errors } else { Write-Host ($file.Name + " removed") }

}

}

} #DeletePerFormat

3.1 function ListUsers

{

param ($filter)

Clear-Host

Header "Users"

#All users

$users = Get-LocalUser

#Account or ~~password~~ expired if ($filter -eq "Expired")

{

$users = $users | Where-Object { ( $\_.AccountExpires -ne $null `

-and $\_.AccountExpires -lt (get-date) )`

-or ( $\_.PasswordExpires -ne $null `

-and $\_.Passwordexpires -lt (get-date) )`

}

}

#DescriptIon TPG527C

if ($filter -eq "TPG527C")

{

$users = $users | Where-Object { $\_.Description -eq "TPG527C" }

}

#Select properties and display

$users | select-object Name, LastLogon, AccountExpires, PasswordExpires `

| Sort Name `

| Format-Table

} #ListUsers

3.2 function AddUser

{

Clear-Host

Header "Add user"

$username = Read-Host "User name"

$password = Read-Host -prompt "Password" -AsSecureString

$exists = (get-localuser).name -contains $username if (!$exists)

{

$today = get-date

$accountExpires = $today.AddDays(-1)

new-localuser -name $username ` -password $password `

-AccountExpires $accountExpires `

-Description "TPG527C" write-host "User " $username " created."

Add-LocalGroupMember -Group Users -Member $username write-host "User " $username " added to group " Users + "."

} #does not exists

}#Add user

4.1 function CPU

{

Clear-Host Header "CPU Details"

get-ciminstance win32\_processor `

| format-list

} #CPU

4.2 function OperatingSystem

{

Clear-Host

Header "Operating system"

Get-ComputerInfo "OS\*"

} #OperatingSystem

4.3 function Printers

{

ClearScreen

Header "Installed printers" get-ciminstance win32\_printer ` | select-object Name, Drivername `

| Format-Table

} #Printers

4.4 function DiskDrives

{

Clear-Host

Header "Disk Drives"

get-ciminstance win32\_logicaldisk `

| Select-Object `

Name,

FileSystem,

@{N="Size (GB)"; E = { [Math]::Round($\_.Size / 1GB)}},

@{N="Free (GB)"; E = { [Math]::Round($\_.FreeSpace / 1GB)}} `

| Format-Table

} #DiskDrives

5.1 function Main

{

#6.1 Create connection

$pw = convertto-securestring

-asplaintext

-force

-string "Remote"

$cred = new-object

-typename System.management.Automation.PSCredential

-argumentlist "Remote", $pw

$s = new-pssession

-computername Remote

-credential $cred

#Loop through menu items do

{

$option = Menu

switch ($option)

{

"XO" { Invoke-Command -Session $s `

5.2 -scriptblock { Start-Process Explorer } `

}

"XX" { Invoke-Command -Session $s -scriptblock ${function:CloseExplorer} }

"C" { (CopyToRemote $s) }

"L" { (ListFiles $s) }

}

Read-Host "Press Enter to return to the menu"

} until ($option -eq "X")

} #Main

5.3 function CloseExplorer

{

$ID = (get-process `

| Where-Object {$\_ -like "\*Explorer\*"} `

).Id

if ( !($ID -eq $null))

{

stop-process -Id $ID

}

} #ExplorerClose

5.4 function CopyToRemote {

param([object] $s)

Copy-Item `

"C:\TPG527C\_Test3" `

-Destination "C:\" `

-recurse `

-ToSession $s

} #CopyToRemote

5.5 function ListFiles {

param([object] $s) Clear-Host

Header "List files"

invoke-command -Session $s `

-scriptblock `

{

get-childitem -Path "C:\TPG527C\_Test3"

}

} #ListFiles