Q1. #include <EditConstants.au3>

#include <GUIConstantsEx.au3>

#include <StaticConstants.au3>

#include <WindowsConstants.au3>

; Create the GUI

GUICreate("Calculator - Samiksha Khot 29", 260, 230)

; Digit buttons

Local $idBtn0 = GUICtrlCreateButton("0", 54, 171, 36, 29)

Local $idBtn1 = GUICtrlCreateButton("1", 54, 138, 36, 29)

Local $idBtn2 = GUICtrlCreateButton("2", 93, 138, 36, 29)

Local $idBtn3 = GUICtrlCreateButton("3", 132, 138, 36, 29)

Local $idBtn4 = GUICtrlCreateButton("4", 54, 106, 36, 29)

Local $idBtn5 = GUICtrlCreateButton("5", 93, 106, 36, 29)

Local $idBtn6 = GUICtrlCreateButton("6", 132, 106, 36, 29)

Local $idBtn7 = GUICtrlCreateButton("7", 54, 73, 36, 29)

Local $idBtn8 = GUICtrlCreateButton("8", 93, 73, 36, 29)

Local $idBtn9 = GUICtrlCreateButton("9", 132, 73, 36, 29)

Local $idBtnPeriod = GUICtrlCreateButton(".", 132, 171, 36, 29)

; Memory buttons

Local $idBtnMClear = GUICtrlCreateButton("MC", 8, 73, 36, 29)

Local $idBtnMRestore = GUICtrlCreateButton("MR", 8, 106, 36, 29)

Local $idBtnMStore = GUICtrlCreateButton("MS", 8, 138, 36, 29)

Local $idBtnMAdd = GUICtrlCreateButton("M+", 8, 171, 36, 29)

; Operator buttons

Local $idBtnChangeSign = GUICtrlCreateButton("+/-", 93, 171, 36, 29)

Local $idBtnDivision = GUICtrlCreateButton("/", 171, 73, 36, 29)

Local $idBtnMultiplication = GUICtrlCreateButton("\*", 171, 106, 36, 29)

Local $idBtnSubtract = GUICtrlCreateButton("-", 171, 138, 36, 29)

Local $idBtnAdd = GUICtrlCreateButton("+", 171, 171, 36, 29)

Local $idBtnAnswer = GUICtrlCreateButton("=", 210, 171, 36, 29)

Local $idBtnInverse = GUICtrlCreateButton("1/x", 210, 138, 36, 29)

Local $idBtnSqrt = GUICtrlCreateButton("Sqrt", 210, 73, 36, 29)

Local $idBtnPercentage = GUICtrlCreateButton("%", 210, 106, 36, 29)

; Control buttons

Local $idBtnBackspace = GUICtrlCreateButton("Backspace", 54, 37, 63, 29)

Local $idBtnClearE = GUICtrlCreateButton("CE", 120, 37, 62, 29)

Local $idBtnClear = GUICtrlCreateButton("C", 185, 37, 62, 29)

; Edit screen and memory label

Local $idEdtScreen = GUICtrlCreateEdit("0.", 8, 2, 239, 23, BitOR($ES\_READONLY, $ES\_RIGHT), $WS\_EX\_STATICEDGE)

Local $idLblMemory = GUICtrlCreateLabel("", 12, 39, 27, 26, $SS\_SUNKEN)

GUISetState()

; Event loop

Local $msg

Do

$msg = GUIGetMsg()

Switch $msg

Case $GUI\_EVENT\_CLOSE

Exit

; Add your button actions here later, for example:

Case $idBtn0

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "0")

Case $idBtn1

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "1")

Case $idBtn2

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "2")

Case $idBtn3

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "3")

Case $idBtn4

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "4")

Case $idBtn5

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "5")

Case $idBtn6

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "6")

Case $idBtn7

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "7")

Case $idBtn8

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "8")

Case $idBtn9

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & "9")

Case $idBtnPeriod

GUICtrlSetData($idEdtScreen, GUICtrlRead($idEdtScreen) & ".")

; Add more cases for each button for their respective functionalities

EndSwitch

Until $msg = $GUI\_EVENT\_CLOSE

Q3)file path finder

include <FileConstants.au3>

#include <MsgBoxConstants.au3>

Example()

Func Example()

; Create a constant variable in Local scope of the message to display in FileOpenDialog.

Local Const $sMessage = "Select a single file of any type."

; Display an open dialog to select a file.

Local $sFileOpenDialog = FileOpenDialog($sMessage, @WindowsDir & "\", "All (.)", $FD\_FILEMUSTEXIST)

If @error Then

; Display the error message.

MsgBox($MB\_SYSTEMMODAL, "", "No file was selected.")

; Change the working directory (@WorkingDir) back to the location of the script directory as FileOpenDialog sets it to the last accessed folder.

FileChangeDir(@ScriptDir)

Else

; Change the working directory (@WorkingDir) back to the location of the script directory as FileOpenDialog sets it to the last accessed folder.

FileChangeDir(@ScriptDir)

; Replace instances of "|" with @CRLF in the string returned by FileOpenDialog.

$sFileOpenDialog = StringReplace($sFileOpenDialog, "|", @CRLF)

; Display the selected file.

MsgBox($MB\_SYSTEMMODAL, "", "You chose the following file:" & @CRLF & $sFileOpenDialog)

EndIf

EndFunc ;==>Example

; Q4: Write a script that generates a log file whenever it's run, recording the current date and time and appending a message.

; -------------------------------------------------------

$file = "script\_log.txt"

FileWrite($file, @YEAR & "-" & @MON & "-" & @MDAY & " " & @HOUR & ":" & @MIN & ":" & @SEC & " - Script executed successfully." & @CRLF)

MsgBox(0, "Log Updated", "Log updated successfully.")

; Q5: Write a script to identify and move all .lnk (shortcut) files from the desktop to a folder called "Shortcuts" on the desktop.

#include <File.au3>

Local $desktopPath = @DesktopDir

Local $shortcutFolder = $desktopPath & "\Shortcut"

If Not FileExists($shortcutFolder) Then

DirCreate($shortcutFolder)

EndIf

Local $files = \_FileListToArray($desktopPath,".", 1)

If Not @error Then

For $i = 1 To $files[0]

Local $sourceFile = $desktopPath & "\" & $files[$i]

Local $destinationFile = $shortcutFolder & "\" & $files[$i]

FileMove($sourceFile, $destinationFile, 9)

Next

Else

MsgBox(64,"Info","No files found on the desktop.")

EndIf

MsgBox(64,"Done","All files have been moved to the 'Shortcut' folder.")

Q6. ; Q6: Write a script that identifies a specific window (e.g., Notepad) by its title and resizes it to specific dimensions.

; -------------------------------------------------------

Local $windowTitle = "Untitled - Notepad"

Local $width = 800

Local $height = 200

Run("notepad.exe")

If WinWait($windowTitle, "", 5) Then

WinWaitActive($windowTitle)

WinMove($windowTitle, "", Default, Default, $width, $height)

MsgBox(0, "Success", "Notepad has been opened and resized to " & $width & "x" & $height & ".")

Else

MsgBox(16, "Error", "The Notepad window could not be opened or found.")

EndIf

Q7: Write a script that opens a specified .txt file, reads its contents, and displays them in a message box or GUI.

; ------------------------------------------------------

#include <GUIConstantsEx.au3>

Global Const $ES\_AUTOVSCROLL = 0x0040

Global Const $ES\_MULTILINE = 0x0004

Global Const $WS\_VSCROLL = 0x00200000

Global $filePath = FileOpenDialog("Select a Text File", @ScriptDir, "Text Files (\*.txt)", 1)

If @error Then

MsgBox(16, "Error", "No file selected. Exiting script.")

Exit

EndIf

Local $fileHandle = FileOpen($filePath, 0)

If $fileHandle = -1 Then

MsgBox(16, "Error", "Failed to open the file. Exiting script.")

Exit

EndIf

Local $fileContents = FileRead($fileHandle)

FileClose($fileHandle)

If $fileContents = "" Then

MsgBox(48, "Info", "The file is empty.")

Exit

EndIf

Local $gui = GUICreate("File Contents Viewer", 600, 400)

Local $edit = GUICtrlCreateEdit($fileContents, 10, 10, 580, 380)

BitOR($ES\_AUTOVSCROLL, $ES\_MULTILINE, $WS\_VSCROLL)

GUICtrlSetFont($edit, 10, 400, 0, "Consolas")

GUISetState(@SW\_SHOW, $gui)

While True

Switch GUIGetMsg()

Case $GUI\_EVENT\_CLOSE

ExitLoop

EndSwitch

WEnd

GUIDelete($gui)

Exit

Q8: Write a script that takes user input to specify a folder name and creates the folder in a predefined location.

; -------------------------------------------------------

Global $predefinedLocation = @DesktopDir ; Change this to your desired path

; Display an input box to get the folder name from the user

Local $folderName = InputBox("Create Folder", "Enter the name of the folder to create:", "", "", 300, 150)

; Check if the user provided a name

If @error = 1 Then

MsgBox(16, "Operation Cancelled", "No folder name provided. Exiting script.")

Exit

EndIf

; Construct the full folder path

Local $folderPath = $predefinedLocation & "\" & $folderName

; Check if the folder already exists

If FileExists($folderPath) Then

MsgBox(48, "Folder Exists", "The folder '" & $folderName & "' already exists in the location: " & $predefinedLocation)

Else

; Attempt to create the folder

If DirCreate($folderPath) Then

MsgBox(64, "Success", "The folder '" & $folderName & "' was successfully created at: " & $predefinedLocation)

Else

MsgBox(16, "Error", "Failed to create the folder. Please check the folder name or permissions.")

EndIf

EndIf

; Q10: Write a script that asks the user for a phrase and a number, then repeatedly types the phrase that many times into any open application.

; -------------------------------------------------------

; Ask the user for the phrase to repeat

Local $phrase = InputBox("Text Repeater", "Enter the phrase to repeat:")

; Ask the user for the number of repetitions

Local $repeatCount = InputBox("Text Repeater", "How many times to repeat?", "1")

; Validate inputs

If $phrase = "" Or $repeatCount = "" Then Exit ; Exit if inputs are empty

If Not StringIsInt($repeatCount) Then Exit ; Exit if repetitions are not a number

; Inform the user and provide a delay to switch applications

MsgBox(64, "Get Ready", "Focus on the application. Typing starts in 3 seconds.")

Sleep(3000) ; 3-second delay

; Repeat the text

For $i = 1 To $repeatCount

Send($phrase & "{ENTER}") ; Type the phrase and press Enter

Sleep(100) ; Short delay between repetitions

Next

; Q11: Write a script that calculates the total size of a specified folder and displays the result in a message box.

; -------------------------------------------------------

#include <File.au3>

; Prompt user to select a folder

Global $folderPath = FileSelectFolder("Select a Folder", "")

; Check if the user selected a folder

If @error Then

MsgBox(16, "Error", "No folder selected. Exiting script.")

Exit

EndIf

; Function to calculate folder size

Func GetFolderSize($sFolder)

Local $iSize = 0

Local $aFileList = \_FileListToArray($sFolder, "\*", 1)

If IsArray($aFileList) Then

For $i = 1 To $aFileList[0]

$iSize += FileGetSize($sFolder & "\" & $aFileList[$i])

Next

EndIf

Local $aFolderList = \_FileListToArray($sFolder, "\*", 2)

If IsArray($aFolderList) Then

For $i = 1 To $aFolderList[0]

$iSize += GetFolderSize($sFolder & "\" & $aFolderList[$i])

Next

EndIf

Return $iSize

EndFunc

; Get the total size of the selected folder

Local $totalSize = GetFolderSize($folderPath)

; Convert size to human-readable format

Func ConvertSize($size)

If $size < 1024 Then

Return $size & " Bytes"

ElseIf $size < 1048576 Then

Return Round($size / 1024, 2) & " KB"

ElseIf $size < 1073741824 Then

Return Round($size / 1048576, 2) & " MB"

Else

Return Round($size / 1073741824, 2) & " GB"

EndIf

EndFunc

; Display the result

MsgBox(64, "Folder Size", "The total size of the folder is: " & ConvertSize($totalSize))

; Q13: Write a script that checks if a specific process (e.g., notepad.exe) is running. If it is, display a message; otherwise, start the application.

; -------------------------------------------------------

Global $sProcessName = InputBox("Process Checker", "Enter the process name to check (e.g., notepad):")

; Exit if no input is provided

If $sProcessName = "" Then

MsgBox(48, "Process Checker", "No process name provided. Exiting...")

Exit

EndIf

; Ensure the process name has the .exe extension

If StringRight($sProcessName, 4) <> ".exe" Then

$sProcessName &= ".exe"

EndIf

; Check if the process is running

If ProcessExists($sProcessName) Then

MsgBox(64, "Process Checker", $sProcessName & " is already running.")

Else

MsgBox(48, "Process Checker", $sProcessName & " is not running. Starting it now...")

; Try to start the process (removing .exe for compatibility)

Run(StringTrimRight($sProcessName, 4))

EndIf

; Q14: Write a script that deletes all files in the system's temporary folder and displays a summary of the number of files deleted.

; -------------------------------------------------------

Local $tempFolder = @TempDir

Local $deletedCount = 0

; Get the list of files and folders

Local $search = FileFindFirstFile($tempFolder & "\\*")

If $search = -1 Then

MsgBox(16, "Temporary File Cleaner", "Failed to access the temp folder.")

Exit

EndIf

; Loop through all files and folders

While 1

Local $file = FileFindNextFile($search)

If @error Then ExitLoop

Local $filePath = $tempFolder & "\" & $file

; Check if it's a folder

If StringInStr(FileGetAttrib($filePath), "D") Then

DirRemove($filePath, 1) ; Recursively delete folders

Else

If FileDelete($filePath) Then

$deletedCount += 1

EndIf

EndIf

WEnd

FileClose($search)

MsgBox(64, "Temporary File Cleaner", $deletedCount & " files deleted.")

; Q15: Write a script with an input box where the user can specify the number of minutes after which the system should shut down.

; -------------------------------------------------------

#include <GUIConstantsEx.au3>

; Create GUI

$gui = GUICreate("Custom Shutdown Timer", 300, 150)

GUICtrlCreateLabel("Enter shutdown time in minutes:", 20, 20, 250, 20)

$input = GUICtrlCreateInput("", 20, 50, 100, 20)

$shutdownButton = GUICtrlCreateButton("Schedule Shutdown", 20, 90, 120, 30)

$cancelButton = GUICtrlCreateButton("Cancel Shutdown", 150, 90, 120, 30)

GUISetState(@SW\_SHOW)

While 1

$msg = GUIGetMsg()

Select

Case $msg = $GUI\_EVENT\_CLOSE

ExitLoop

Case $msg = $shutdownButton

; Get the input time and validate

$minutes = GUICtrlRead($input)

If StringIsInt($minutes) And $minutes > 0 Then

; Convert minutes to seconds

$seconds = $minutes \* 60

; Schedule shutdown

Run(@ComSpec & " /c shutdown -s -t " & $seconds, "", @SW\_HIDE)

MsgBox(64, "Shutdown Scheduled", "System will shut down in " & $minutes & " minute(s).")

Else

MsgBox(16, "Invalid Input", "Please enter a valid positive integer.")

EndIf

Case $msg = $cancelButton

; Cancel any scheduled shutdown

Run(@ComSpec & " /c shutdown -a", "", @SW\_HIDE)

MsgBox(64, "Shutdown Cancelled", "Any scheduled shutdown has been canceled.")

EndSelect

WEnd

; Cleanup

GUIDelete($gui)

Exit

; Q16: Write a script for a GUI version of Notepad.

; -------------------------------------------------------

#include <GUIConstantsEx.au3>

#include <EditConstants.au3> ; Required for $ES\_MULTILINE

#include <WindowsConstants.au3> ; Required for $WS\_VSCROLL

Global $hGUI, $hEdit

; Create GUI Window

$hGUI = GUICreate("Simple Notepad", 500, 400)

; Create Edit Box

$hEdit = GUICtrlCreateEdit("", 10, 10, 480, 340, $ES\_MULTILINE + $WS\_VSCROLL)

; Create Menu

$hMenuFile = GUICtrlCreateMenu("File")

$hOpen = GUICtrlCreateMenuItem("Open", $hMenuFile)

$hSave = GUICtrlCreateMenuItem("Save", $hMenuFile)

GUICtrlCreateMenuItem("", $hMenuFile) ; Separator

$hExit = GUICtrlCreateMenuItem("Exit", $hMenuFile)

$hMenuEdit = GUICtrlCreateMenu("Edit")

$hCut = GUICtrlCreateMenuItem("Cut", $hMenuEdit)

$hCopy = GUICtrlCreateMenuItem("Copy", $hMenuEdit)

$hPaste = GUICtrlCreateMenuItem("Paste", $hMenuEdit)

GUICtrlCreateMenuItem("", $hMenuEdit) ; Separator

$hClear = GUICtrlCreateMenuItem("Clear", $hMenuEdit)

$hMenuView = GUICtrlCreateMenu("View")

$hWordWrap = GUICtrlCreateMenuItem("Toggle Word Wrap", $hMenuView)

GUISetState(@SW\_SHOW)

While 1

Switch GUIGetMsg()

Case $GUI\_EVENT\_CLOSE, $hExit

Exit

Case $hOpen, $hSave, $hCut, $hCopy, $hPaste, $hClear, $hWordWrap

EndSwitch

Wend

q.Screenshotprtsce

#include <ScreenCapture.au3>

; Set Hotkeys

HotKeySet("^!s", "TakeScreenshot") ; Ctrl + Alt + S ? Capture Screenshot

HotKeySet("^!e", "ExitScript") ; Ctrl + Alt + E ? Exit Script

; Keep script running

While True

Sleep(100)

WEnd

; Screenshot Function

Func TakeScreenshot()

Local $saveFolder = "C:\screenshot"

; Create folder if it doesn’t exist

If Not FileExists($saveFolder) Then DirCreate($saveFolder)

; Generate filename with timestamp

Local $timestamp = @YEAR & @MON & @MDAY & "\_" & @HOUR & @MIN & @SEC

Local $filePath = $saveFolder & "\Screenshot\_" & $timestamp & ".png"

; Capture the entire screen and save

\_ScreenCapture\_Capture($filePath)

; Show a notification for 3 seconds

TrayTip("Screenshot Taken!", "Saved: " & $filePath, 3000)

EndFunc

; Exit Script Function

Func ExitScript()

Exit

EndFunc

Q2.hotkeys

; Bind hotkeys to specific actions

HotKeySet("^!n", "OpenNotepad") ; Ctrl + Alt + N to open Notepad

HotKeySet("^!b", "OpenFolder") ; Ctrl + Alt + B to open a folder

HotKeySet("^!e", "ExitScript") ; Ctrl + Alt + E to exit the script

; Keep the script running

While 1

Sleep(100)

WEnd

; Function to open Notepad

Func OpenNotepad()

Run("notepad.exe")

EndFunc

; Function to open a specific folder

Func OpenFolder()

Run("explorer.exe C:\Your\Folder\Path") ; Replace with your desired folder path

EndFunc

; Function to exit the script

Func ExitScript()

Exit

EndFunc