# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

# Script Name: Blackjack.ps1 (The Blackjack Game)

# Version: 1.0

# Author: Jerry Lee Ford, Jr.

# Date: January 1, 2007

#

# Description: This PowerShell script is a single player implementation of

# the popular casino blackjack game

#

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Initialization Section

$startGame = "False" #Variable used to determine if the game is played

$playerBusted = "False" #Variable used to track when the player busts

$randomNo = New-Object System.Random #This variable stores a random object

$playerHand = 0 #Stores the current value of the player's hand

$computerHand = 0 #Stores the current value of the computer's hand

$playAgain = "True" #control the execution of the loop that controls the

#execution of logic in the main processing section

# Functions and Filters Section

#This function gets the player's permission to begin the game

function Get-Permission {

#Loop until a valid reply is collected

while ($startGame -eq "False") {

Clear-Host #Clear the Windows command console screen

#Display the game's opening screen

Write-Host "`n`n`n"

Write-Host " Welcome to the" -foregroundColor Blue

write-Host ""

Write-Host ""

Write-Host " P O W E R S H E L L B L A C K J A C K G A M E"`

-foregroundColor Blue

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

#Collect the player's input

$response = Read-Host "`n`n`n`n`n`n`n Would you like to play? (Y/N)"

#Validate the player's input

if ($response -eq "Y"){ #The player wants to play

$startGame = "True"

}

elseif ($response -eq "N") { #The player wants to quit

Check-Registry

exit #Terminate script execution

}

}

}

#This function retrieves a registry value that specifies whether or not

#the script should display a splash screen if the player chooses not to

#play a game after starting the script

function Check-Registry {

Clear-Host #Clear the Windows command console screen

$currentLocation = Get-Location #Keep track of the current directory

set-Location hkcu:\ #Change to the HKEY\_CURRENT\_USER hive

#Retrieve the data stored in the Credits value under the PSBlackjack

#subkey

$regKey = $(Get-ItemProperty hkcu:\PSBlackjack).Credits

if ($regKey -eq "True") { #If the registry value is set to true

#display the closing splash screen

Write-Host " `n`n`n"

Write-Host " P O W E R S H E L L B L A C K J A C K`n`n`n"`

-foregroundColor Blue

write-Host " Developed by Jerry Lee Ford, Jr.`n`n"

Write-Host " Copyright 2007`n`n`n`n"

Write-Host " www.tech-publishing.com`n`n`n`n`n`n"

}

Set-Location $currentLocation #Restore the current working directory

}

#This function controls the execution of an individual round of play

function Play-Game {

Deal-Hand #Call the function that deals the opening hands

Get-PlayerHand #Call the function that manages the player's hand

#If the player has busted the game is over, otherwise it is the

#computer's turn

if ($script:playerBusted -eq "False") {

Get-ComputerHand #Call the function that manages the computer's hand

}

Analyze-Results #Call the function that analyzes game results and

#declares a winner

}

#This function deals the player and computer's initial hands

function Deal-Hand {

$script:playerHand = Get-Card #Assign a card to the player's hand

$script:computerHand = Get-Card #Assign a card to the computer's hand

}

#This function retrieves a random number representing a card and returns

#the value of that card back to the calling statement

function Get-Card {

$number = 0

#Generate the game's random number (between 1 - 13)

$number = $randomNo.Next(1, 14)

if ($number -eq 1 ) {$number = 11} #Represents an ace

if ($number -gt 10) {$number = 10} #Represents a jack, queen or king

$number #Return the number back to the calling statements

}

#This function is responsible for managing the computer's hand

function Get-ComputerHand {

$tempCard = 0 #Stores the value of the computer's new card

#The computer continues to take hits as long as its hand's value is less

#than seventeen

while ($computerHand -lt 17) {

$tempCard = Get-Card #Get a new card for the computer

#Add the value of the new card to the computer's hand

$script:computerHand = $script:computerHand + $tempCard

}

}

#This function analyzes and displays the results of each game

function Analyze-Results {

Clear-Host #Clear the Windows command console screen

#Display the player and computer's final hand

Write-Host "`n`n`n`n RESULTS:`n`n"

Write-host " Player Hand: $playerHand`n"

Write-Host " Computer Hand: $computerHand`n`n"

#See if the player busted

if ($playerBusted -eq "True") {

Write-Host "`a You have gone bust." -ForegroundColor Blue

}

else { #See if the computer busted

if ($computerHand -gt 21) {

Write-host "`a The computer has gone bust." -ForegroundColor Blue

}

else { #Neither the player nor the computer busted so look for a winner

if ($playerHand -gt $computerHand) {

Write-Host "`a You Win!" -ForegroundColor Blue

}

if ($playerHand -eq $computerHand) {

Write-Host "`a Tie!" -ForegroundColor Blue

}

if ($playerHand -lt $computerHand) {

Write-host "`a You loose." -ForegroundColor Blue

}

}

}

}

#This function displays the value of both the player and computer's

#current hands and prompts the player to take another card

function Get-PlayerHand {

$keepGoing = "True" #Control the execution of the loop that managers

#the player's hand

$response = "" #Stores the players input

#Loop until a valid reply is collected

while ($keepGoing -eq "True") {

Clear-Host #Clear the Windows command console screen

#Display the player and computer's current hands

Write-Host "`n`n"

Write-Host ""

write-Host " CURRENT HAND:"

Write-Host "`n"

Write-Host " Player Hand: $playerHand"

Write-Host ""

Write-Host " Computer Hand: $computerHand"

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

Write-Host ""

#Prompt the player to take another card

$response = Read-Host "`n`n`n`n`n`n`n Do you want another card? (Y/N)"

#Validate the player's input

if ($response -eq "Y"){

Get-NewCard #Get another card for the player

}

elseif ($response -eq "N") { #The player wants to quit

$keepGoing = "False"

Clear-Host #Clear the Windows command console screen

}

if ($playerHand -gt 21) { #The player has gone bust

$script:playerBusted = "True"

$keepGoing = "False"

}

}

}

#This function is called whenever the player elects to get a new card

#and is responsible for updating the value of the player's hand

function Get-NewCard {

$tempCard = 0 #Stores the value of the player's new card

$tempCard = Get-Card #Get a new card for the player

#Add the value of the new card to the player's hand

$script:playerHand = $script:playerHand + $tempCard

}

# Main Processing Section

Get-Permission #Call function that asks the players for permission to

#start the game

#Continue playing new games until the player decides to quit the game

while ($playAgain -eq "True") {

Play-Game #Call function that controls the play of individual games

#Prompt the player to play a new game

$response = Read-Host "`n`n`n`n`n`n`n`n`n`n Press Enter to play"`

"again or Q to quit"

if ($response -eq "Q") { #The player wants to quit

$playAgain = "False"

Clear-Host #Clear the Windows command console screen

}

else { #The player did not enter Q so let's keep playing

$playAgain = "True"

$playerBusted = "False"

}

}