# Tutorial #5 – Pointers

## Overview

This tutorial will show a few examples of how to create achievements for a game that uses \*pointers\*. (Spider-Man)[https://retroachievements.org/game/11319] was chosen for the tutorial because it uses several pointers for Spider-Man himself and other characters in the game. If you are unfamiliar with \*pointers\* then please review the (Pointer/Add Address Tutorial video)[https://youtu.be/\_gk0vYYlm-E] by (wilhitewarrior)[https://retroachievements.org/user/wilhitewarrior]. The video contains a great explanation of pointers and how to find them.<br>



!(Spider-Man Title Screen)[Spiderman\_Title.png]

## Pointers

\*Pointers\* are variables that point to another memory address. They are commonly used for dynamic memory where the program will ask for a specified amount ofmemory and the operating system will allocate the next available block of memory to it. The \*pointer\* is then assigned the memory address for the newly created memory. Once the program is done with the memory it is returned to the operating system. Because of the dynamic nature of acquiring and releasing memory a \*pointer\* will rarely point to the same address.

Dynamic memory is a modern feature so you’re more likely to run into \*pointers\* on newer systems like the N64, Playstation, or Dreamcast although they do show up occasionally on earlier systems. Pointers are often 32-bits with the address being in the bottom 24-bits and the top 8-bits commonly being 0x80, 0x90, 0xc0, or 0xff (depends on the underlying operating system). We can ignore the top 8-bits and use the bottom 24-bits to figure out where the memory is located. A 24-bit \*pointer\* can address up to 16MB of memory so later generation consoles may use the full 32-bits to represent higher memory addresses and likewise older generation consoles may have 16-bit \*pointers\*.

## Data Structures

\*Data structures\* are a specialized way for organizing memory in object oriented programming. \*Data structures\* are used to group pieces of data together and are sometimes organized to improve performance. Consider the \*data structure\* used for Spider-Man’s player stats below:

```

Player Stats Data Structure

+04 - X Coordinates

+08 - Y Coordinates

+0C - Z Coordinates

+DE - Health

+140 - Animation ID

+150 - Another Animation ID?

+1A8 - Cutscene Playing (01)

+5D8 - Webbing

+5DC - Web Cartridges

+5F0 - Spidey Armor Active

+5F8 - Spidey Armor Strength

+5EC - Magnesium Webbing (01)

+EE8 - Pointer to Current Enemy

+EEB - Target Type

+1014 - Max Health

```

Note that the \*data structure\* is incomplete and has many more variables that were not required for the achievements. When assigning a new \*pointer\* the address points to the root of the data structure at \*data offset\* 0x0. If we want to know the address for Spider-Man’s health we would need to add 0xDE to the player stats \*pointer\*. Similarly, if we want to know who Spider-Man is targeting we would add 0xEE8 to the player stats \*pointer\*. In this case we would have a double \*pointer\*, a \*pointer\* that points to another \*pointer\*, which is shown in (Example 5B)[Example5B.md]. One of the nice things about \*data structures\* is that since they group data together once you find a useful memory address there will likely be other useful memory addresses around it.

# Finding Pointers

How do you know if you have a \*pointer\*? \*Pointers\* can change at any time but are usually set during a load sequence. To find a \*pointer\* it’s best to dig memory in multiple game instances. If you notice that a memory address differs between instance that might have a \*pointer\* on your hands. To find a pointer:

1. Make three or more save states and save a note for the memory address of interest in each state. For this example we will be using three states.

```

Mem\_1 = Memory address in state 1

Mem\_2 = Memory address in state 2

Mem\_3 = Memory address in state 3

```

1. Find the address differences between the states by using a [hex calculator](<https://www.calculator.net/hex-calculator.html>). For a three state example:

```

MemDiff1\_2 = Mem\_2 – Mem\_1

MemDiff2\_3 = Mem\_3 – Mem\_2

```

1. Start in state 1 and start a memory filter for a 24-bit address (may be 16-bit for Mega Drive or SNES)
2. Switch to state 2

```

If MemDiff1\_2 > 0 then filter “Last Value Plus” by MemDiff1\_2

If MemDiff1\_2 < 0 then filter “Last Value Minus” abs(MemDiff1\_2)

```

1. Switch to state 3

```

If MemDiff2\_3 > 0 then filter “Last Value Plus” by MemDiff2\_3

If MemDiff2\_3 < 0 then filter “Last Value Minus” by abs(MemDiff2\_3)

```

1. At this point you should have a couple of good \*pointer\* candidates. If not then you may need more than three save states.
2. Select a pointer with the value that is a little \*less\* than the address of the memory your testing. The reason why it should be less is to account for the data structure of the memory ```Candidate = value of the \*pointer\* candidate```.
3. Figure out the \*data offset\* for the current memory by ```Data\_Offset = Mem\_3 - Candidate```
4. Make a small test achievement using the new \*pointer\* and \*data offset\* to check if it works for all the states. If we are testing a byte the syntax would be ```byte(tbyte(Candidate) + Data\_Offset)```

# Example 5A: Get Over Here!

Spider-Man has an impressive rogues gallery of animal based villains that he must stop to protect the citizens of New York. In this achievement Scorpion is trying to settle a score with J. Jonah Jameson and Spider-Man is the only one who can save him. As a challenge the web-slinging Spider-Man must defeat Scorpion without taking damage, letting Jameson take damage, or letting Scorpion find Jameson’s hiding place.<br>



!(Spider-Man Vs. Scorpion)[Spiderman\_Vs\_Scorpion.png]

```

// Spider-Man

// #ID = 11319

// $0A568D: Chapter ID (US)

function ChapterID() => byte(0x0A568D)

// $0A568F: Level ID? (US)

function LevelID() => byte(0x0A568F)

// $0A5704: Costume Selected (US)

// 00 - Default

// 01 - Spider-Man 2099

// 02 - Symbiote Spider-Man

// 03 - Captain Universe

// 04 - Spider-Man Unlimited

// 05 - Amazing Bag-Man

// 06 - Scarlet Spider-Man

// 07 - Ben Reilly

// 08 - Quick-Change Spider-Man

// 09 - Peter Parker

function CostumeSelected() => byte(0x0A5704)

// $0B4F34: Various Flags (US)

// 01 - Menu

// 02 - Game Over

// 03 - Level Completed Flag

// 07 - Quit to Main Menu

// 08 - Level Restart

// 0A - Quit to Training

function Flags() => byte(0x0B4F34)

// $0B4F38: Level Time? (32-Bit) (US)

// (Continues counting while paused)

function LevelTime() => dword(0x0B4F38)

// $0B4F40: Difficulty? (US)

// 00 - Kid Mode

// 01 - Easy

// 02 - Normal

// 03 - Hard

function Difficulty() => byte(0x0B4F40)

// $0B4F6C: Invincibility Cheat Boolean (00/01) (US)

function InvincibilityCheat() => byte(0x0B4F6C)

// $0B4F98: Unlimited Webbing Cheat Boolean (00/01) (US)

function UnlimitedWebbingCheat() => byte(0x0B4F98)

// $0B5268: Player Stats Pointer? (US)

// +04 - X Coordinates

// +08 - Y Coordinates

// +0C - Z Coordinates

// +DE - Health

// +140 - Animation ID

// +150 - Another Animation ID?

// +1A8 - Cutscene Playing (01)

// +5D8 - Webbing

// +5DC - Web Cartridges

// +5F0 - Spidey Armor Active

// +5F8 - Spidey Armor Strength

// +5EC - Magnesium Webbing (01)

// +EE8 - Pointer to Current Enemy

// +EEB - Target Type

// +1014 - Max Health

function PlayerStatsPointer() => tbyte(0x0B5268)

function PlayerStatsNotNull() => dword(0x0B5268) != 0

function PlayerHealth() => byte(PlayerStatsPointer() + 0x0000DE)

// $0B5640: Jameson Health/Stats Pointer

// +DE - Health

// +140 - Animation ID

function JamesonPointer() => tbyte(0x0B5640)

function JamesonNotNull() => dword(0x0B5640) != 0

function JamesonHealth() => byte(JamesonPointer() + 0x0000DE)

function JamesonAnimation() => byte(JamesonPointer() + 0x000140)

// $0B58D8: Cutscene Letterbox Size (1E) (US)

function CutscreenSize() => byte(0x0B58D8)

// No cheating conditions common for the entire set.

function NoCheats()

{

// Pause if any the costume with special abilities are selected

// ie. Spider-Man 2099, Symbiote Spider-Man, Captain Universe, Spider-Man Unlimited

noSpecialCostumes = unless((CostumeSelected() >= 1 && CostumeSelected() <= 4))

// Pause if the player is in kid mode (difficulty 0)

noKidsMode = unless(Difficulty() == 0)

// Pause if the player is cheating

noCheats = unless(InvincibilityCheat() != 0) &&

unless(UnlimitedWebbingCheat() != 0)

return noSpecialCostumes && noKidsMode && noCheats

}

//---------------------------------------------------------------------

// Function to build the "Example 5A: Get Over Here!" achievement

function ScorpionBossFight(title, description, points)

{

// Trigger achievement when we are on the level Spidey Vs. Scorpion! and

// the level complete flag turns on

progress = trigger\_when((

prev(ChapterID()) == 50 &&

prev(LevelID()) == 50 &&

Flags() != prev(Flags()) &&

Flags() == 3))

// Add a measure for how long until Jameson is found

start = measured(once(prev(LevelTime()) < 32))

// Reset if the players health decreases

spidyDamageless = never(

PlayerStatsNotNull() &&

PlayerHealth() < prev(PlayerHealth()))

// Reset if Jameson's health decreases

jamesonDamageless = never(

JamesonNotNull() &&

JamesonHealth() < prev(JamesonHealth()))

// Reset if Jameson's is found (Jameson animation changes)

jamesonNotFound = never(

CutscreenSize() == 0 &&

JamesonNotNull() &&

JamesonAnimation() != prev(JamesonAnimation()) &&

JamesonAnimation() == 59)

// Reset if the level changes

noLevelChange = never(prev(ChapterID()) != 50) &&

never(prev(LevelID()) != 50)

achievement(title, description, points, progress && start && spidyDamageless &&

jamesonDamageless && jamesonNotFound && noLevelChange && NoCheats())

}

ScorpionBossFight("Example 5A: Get Over Here!", "Defeat Scorpion without Spidey or Jameson taking damage and without letting Scorpion find Jameson", 25)

```

## Pointer Functions

```

function PlayerStatsPointer() => tbyte(0x0B5268)

function PlayerHealth() => byte(PlayerStatsPointer() + 0x0000DE)

```

To make it easier to reference \*pointers\* you can write helper functions that handle the details. The function ```PlayerStatsPointer()``` references the \*pointer\* address as a ```tbyte``` to use the bottom 24-bits of the address. The function ```PlayerHealth()``` adds the offset ```0xDE``` to the \*pointers’\* address so it dynamically changes it’s address whenever the \*pointer\* changes. Helper functions like these can be made for each \*data offset\* you want to reference.

## Null Pointers

```

function PlayerStatsNotNull() => dword(0x0B5640) != 0

```

A \*pointer\* will be null (zero) before memory is allocated to it or after the memory has been released back to the operating system. Its standard programming practice to check if a \*pointer\* is null before trying to access the memory it is pointing to. It is also good practice to do the same when writing achievements otherwise you could be referencing the wrong memory. For example, if the ```PlayerStatsPointer()``` was null the ```PlayerHealth()``` would be pointed to ```byte(0xDE)```. To avoid this the function ```PlayerStatsNotNull()``` is used with the condition ``` PlayerStatsNotNull() && PlayerHealth() < prev(PlayerHealth()))``` to reset if a player has lost health. Using a null check before referencing a pointer will save you many future headaches.

# Example 5B: I'm Not the Bad Guy!

Despite his heroism, Spider-Man is a vigilante so he often finds himself in trouble with the law. To make matters worse, someone impersonated Spider-Man during a recent heist making him public enemy #1. In this achievement the police are chasing Spider-Man over a series of five different levels. Despite being hunted Spider-Man is a moral superhero who believes that “with great power comes great responsibility” so he must evade the police without harming them.<br>



!(Spider-Man Vs. the Police)[Spiderman\_Vs\_Police.png]

```

// Spider-Man

// #ID = 11319

// $0A568D: Chapter ID (US)

function ChapterID() => byte(0x0A568D)

// $0A568F: Level ID? (US)

function LevelID() => byte(0x0A568F)

// $0A56E4: Level Completed Count: Police Chopper Chase (US)

function PoliceChopperChaseComplete() => byte(0x0A56E4)

// $0A56E5: Level Completed Count: Missile Attack (US)

function MissileAttackComplete() => byte(0x0A56E5)

// $0A56E6: Level Completed Count: Building Top Chase (US)

function BuildingTopChaseComplete() => byte(0x0A56E6)

// $0A56E7: Level Completed Count: Scale the Girders (US)

function ScaleTheGirdersComplete() => byte(0x0A56E7)

// $0A56E8: Level Completed Count: Police Evaded (US)

function PoliceEvadedComplete() => byte(0x0A56E8)

// $0A5704: Costume Selected (US)

// 00 - Default

// 01 - Spider-Man 2099

// 02 - Symbiote Spider-Man

// 03 - Captain Universe

// 04 - Spider-Man Unlimited

// 05 - Amazing Bag-Man

// 06 - Scarlet Spider-Man

// 07 - Ben Reilly

// 08 - Quick-Change Spider-Man

// 09 - Peter Parker

function CostumeSelected() => byte(0x0A5704)

// $0B4F34: Various Flags (US)

// 01 - Menu

// 02 - Game Over

// 03 - Level Completed Flag

// 07 - Quit to Main Menu

// 08 - Level Restart

// 0A - Quit to Training

function Flags() => byte(0x0B4F34)

// $0B4F38: Level Time? (32-Bit) (US)

// (Continues counting while paused)

function LevelTime() => dword(0x0B4F38)

// $0B4F40: Difficulty? (US)

// 00 - Kid Mode

// 01 - Easy

// 02 - Normal

// 03 - Hard

function Difficulty() => byte(0x0B4F40)

// $0B4F6C: Invincibility Cheat Boolean (00/01) (US)

function InvincibilityCheat() => byte(0x0B4F6C)

// $0B4F98: Unlimited Webbing Cheat Boolean (00/01) (US)

function UnlimitedWebbingCheat() => byte(0x0B4F98)

// $0B5268: Player Stats Pointer? (US)

// +04 - X Coordinates

// +08 - Y Coordinates

// +0C - Z Coordinates

// +DE - Health

// +140 - Animation ID

// +150 - Another Animation ID?

// +1A8 - Cutscene Playing (01)

// +5D8 - Webbing

// +5DC - Web Cartridges

// +5F0 - Spidey Armor Active

// +5F8 - Spidey Armor Strength

// +5EC - Magnesium Webbing (01)

// +EE8 - Pointer to Current Enemy

// +EEB - Target Type

// +1014 - Max Health

function PlayerStatsPointer() => tbyte(0x0B5268)

function PlayerStatsNotNull() => dword(0x0B5268) != 0

function PlayerCutscene() => byte(PlayerStatsPointer() + 0x0001A8)

function PlayerTargetType() => dword(PlayerStatsPointer() + 0x000EEB)

function PlayerTargetPointer() => tbyte(PlayerStatsPointer() + 0x000EE8)

function PlayerTargetHealth() => byte(PlayerTargetPointer() + 0x0000DE)

// No cheating conditions common for the entire set.

function NoCheats()

{

// Pause if any the costume with special abilities are selected

// ie. Spider-Man 2099, Symbiote Spider-Man, Captain Universe, Spider-Man Unlimited

noSpecialCostumes = unless((CostumeSelected() >= 1 && CostumeSelected() <= 4))

// Pause if the player is in kid mode (difficulty 0)

noKidsMode = unless(Difficulty() == 0)

// Pause if the player is cheating

noCheats = unless(InvincibilityCheat() != 0) &&

unless(UnlimitedWebbingCheat() != 0)

return noSpecialCostumes && noKidsMode && noCheats

}

// ------------------------------------------------------------------------

// Function to build the "Example 5B: I'm Not the Bad Guy!" achievement

function PoliceChase(title, description, points)

{

// Count the completion of each of the 5 police missions

progress = trigger\_when(

tally(5,

once(PoliceChopperChaseComplete() > prev(PoliceChopperChaseComplete())),

once(MissileAttackComplete() > prev(MissileAttackComplete())),

once(BuildingTopChaseComplete() > prev(BuildingTopChaseComplete())),

once(ScaleTheGirdersComplete() > prev(ScaleTheGirdersComplete())),

once(

prior(ChapterID()) == 51 &&

prior(LevelID()) == 53 &&

PoliceEvadedComplete() > prev(PoliceEvadedComplete()) &&

Flags() == 3

),

always\_false()

)

)

// Add a hit at the beginning of the level

start = measured(

once(

ChapterID() == 51 &&

LevelID() == 49 &&

prev(LevelTime()) < 32

)

)

// Reset if a police officer is harmed

// This is a pointer of a pointer, the player stats pointer has a target pointer in it.

// When the target pointer is pointing to a police officer check if their health decreases.

officerSafe = never(

(PlayerStatsNotNull() &&

(PlayerCutscene() == 0) &&

(PlayerTargetType() == 128) &&

(PlayerTargetPointer() == prev(PlayerTargetPointer())) &&

PlayerTargetHealth() < prev(PlayerTargetHealth()))

)

achievement(title, description, points, progress && start && officerSafe && NoCheats())

}

PoliceChase("Example 5B: I'm Not the Bad Guy!", "Complete all five levels of the police chase without attacking any cops", 10)

```

## Double Pointers

```

function PlayerStatsPointer() => tbyte(0x0B5268)

function PlayerTargetPointer() => tbyte(PlayerStatsPointer() + 0x000EE8)

function PlayerTargetHealth() => byte(PlayerTargetPointer() + 0x0000DE)

```

A major complexity with \*pointers\* is that they can point to another \*pointer\* otherwise known as a double \*pointer\*. You can have a chain of \*pointers\* that continually point to the next or previous piece of memory also known as a linked list. Hopefully you only have to deal with single \*pointers\* however, if you do have double or more pointers you can use the same process described here for each \*pointer\* reference. In the above example the player’s stats \*pointer\* has a target \*pointer\* in it which points to the enemy Spider-Man is looking at. The game uses a red arrow over Spider-Man’s current target. To reference the health of a police officer you’ll need to follow the \*pointer\* from the players’ stats \*pointer\* to the player target \*pointer\*. Fortunately it appears that Spider-Man and the police officers share the same data structure so the police officers health is ```+0xde``` from the address in the player’s target \*pointer\*. The above functions simplify following the \*pointer\* addresses’ so you can use ```PlayerTargetHealth()``` as you would for regular memory.<br>

# Example 5C: No Idea What You're In For

Following the animal theme, Doctor Octopus is a scientist driven mad by the accident that fused four robotic tentacle arms to his body. In this achievement Spider-Man must defeat Doctor Octopus without taking damage or letting Doctor Octopus regenerate his shield two or more times.



!(Spider-Man Vs. Doctor Octopus)[Spiderman\_Vs\_DocOck\_Fight.png]

## Boss Pointer

The boss \*pointer\* at address ```$0B563C``` uses the same data structure as the player stat’s pointer at memory ```$0B5268``` plus a few specific variables used to track events in the fight. Doctor Octopus is using a shield to protect himself from Spider-Man which can be turned off temporary by turning off four switches. The offset ```+40c``` tracks how many of the four switches are currently on. Once all four switches are off the shield will drop and Spider-Man can attack Doctor Octopus directly. After a few seconds the shield will be restored and the switch count will go from zero to four again.

```

+DE - Boss Health

+2F4 - Water Level (Spidey Vs Venom Again)

+40C - Number of switches remaining to deactivate Ock's shield

```

## Homework #5

Create an achievement to defeat Doc Ock without taking damage before his shield can regenerate for a second time.

# Solution #5

Like the player stat pointer the boss pointer should have a null check before using it. It’s worth noting that Spider-Man may have armor at this point so the achievement will reset if Spider-Man loses armor or health. By leveraging the RA scripts the hardest thing about \*pointers\* becomes trying to dig them in memory. Good luck true believers!<br>



!(Spider-Man Vs. Doctor Octopus)[Spiderman\_Vs\_DocOck\_Wins.png]

```

// Spider-Man

// #ID = 11319

// $0A568D: Chapter ID (US)

function ChapterID() => byte(0x0A568D)

// $0A568F: Level ID? (US)

function LevelID() => byte(0x0A568F)

// $0A5704: Costume Selected (US)

// 00 - Default

// 01 - Spider-Man 2099

// 02 - Symbiote Spider-Man

// 03 - Captain Universe

// 04 - Spider-Man Unlimited

// 05 - Amazing Bag-Man

// 06 - Scarlet Spider-Man

// 07 - Ben Reilly

// 08 - Quick-Change Spider-Man

// 09 - Peter Parker

function CostumeSelected() => byte(0x0A5704)

// $0B4F34: Various Flags (US)

// 01 - Menu

// 02 - Game Over

// 03 - Level Completed Flag

// 07 - Quit to Main Menu

// 08 - Level Restart

// 0A - Quit to Training

function Flags() => byte(0x0B4F34)

// $0B4F38: Level Time? (32-Bit) (US)

// (Continues counting while paused)

function LevelTime() => dword(0x0B4F38)

// $0B4F40: Difficulty? (US)

// 00 - Kid Mode

// 01 - Easy

// 02 - Normal

// 03 - Hard

function Difficulty() => byte(0x0B4F40)

// $0B4F6C: Invincibility Cheat Boolean (00/01) (US)

function InvincibilityCheat() => byte(0x0B4F6C)

// $0B4F98: Unlimited Webbing Cheat Boolean (00/01) (US)

function UnlimitedWebbingCheat() => byte(0x0B4F98)

// $0B5268: Player Stats Pointer? (US)

// +04 - X Coordinates

// +08 - Y Coordinates

// +0C - Z Coordinates

// +DE - Health

// +140 - Animation ID

// +150 - Another Animation ID?

// +1A8 - Cutscene Playing (01)

// +5D8 - Webbing

// +5DC - Web Cartridges

// +5F0 - Spidey Armor Active

// +5F8 - Spidey Armor Strength

// +5EC - Magnesium Webbing (01)

// +EE8 - Pointer to Current Enemy

// +EEB - Target Type

// +1014 - Max Health

function PlayerStatsPointer() => tbyte(0x0B5268)

function PlayerStatsNotNull() => dword(0x0B5268) != 0

function PlayerHealth() => byte(PlayerStatsPointer() + 0x0000DE)

function PlayerArmor() => byte(PlayerStatsPointer() + 0x0005F8)

// $0B563C: Boss Pointer (US)

// +DE - Boss Health

// +2F4 - Water Level (Spidey Vs Venom Again)

// +40C - Number of switches remaining to deactivate Ock's shield

function BossPointer() => tbyte(0x0B563C)

function BossNotNull() => dword(0x0B563C) != 0

function BossSwitchesRemaining() => byte(BossPointer() + 0x00040C)

// No cheating conditions common for the entire set.

function NoCheats()

{

// Pause if any the costume with special abilities are selected

// ie. Spider-Man 2099, Symbiote Spider-Man, Captain Universe, Spider-Man Unlimited

noSpecialCostumes = unless((CostumeSelected() >= 1 && CostumeSelected() <= 4))

// Pause if the player is in kid mode (difficulty 0)

noKidsMode = unless(Difficulty() == 0)

// Pause if the player is cheating

noCheats = unless(InvincibilityCheat() != 0) &&

unless(UnlimitedWebbingCheat() != 0)

return noSpecialCostumes && noKidsMode && noCheats

}

// ------------------------------------------------------------------------

// Function to build the "Example 5C: No Idea What You're In For" achievement

function DocOckBossFight(title, description, points)

{

// Trigger the achievement when the level is complete

progress = trigger\_when((Flags() != prev(Flags()) && Flags() == 3))

// Add a hit at the beginning of the level

start = measured(once(prev(LevelTime()) < 32))

// Reset when Dr. Octopus raises the shield twice

noDocOckRegen = never(

repeated(2,

BossPointer() == prev(BossPointer()) &&

!BossNotNull() &&

(prev(BossSwitchesRemaining()) == 0) &&

BossSwitchesRemaining() == 4

)

)

// Reset if the player loses health

spidyDamageless = never((PlayerStatsNotNull() && PlayerHealth() < prev(PlayerHealth())))

// Reset if the player loses armor

spidyNoArmor = never((PlayerStatsNotNull() && PlayerArmor() < prev(PlayerArmor())))

// Reset if game flags are anything other than in game or level complete

gameActive = never((Flags() != 0 && Flags() != 3))

// Reset if we are any level other than Spidey Vs. Dr. Octopus

noLevelChange = never(prev(ChapterID()) != 56) &&

never(prev(LevelID()) != 52)

achievement(title, description, points, progress && start && noDocOckRegen && spidyDamageless &&

spidyNoArmor && gameActive && noLevelChange && NoCheats())

}

DocOckBossFight("Example 5C: No Idea What You're In For", "Defeat Doc Ock without taking damage before his shield can regenerate for a second time", 25)

```