Used in logic to Differentiate nodes			See seperate table below for Character Info node details			On Fresh save in Tough		Possible in Level				
Node Type	Level	Data Type	Code Notes	Size	Final Offset	Nodes from start	Nodes from end	Spawn amount?	Destroy Amount?		Node Type	"+0" "+d8" "+ec"
Character info	0x01					0		A few (unsure why)			Character info	0x111328
Discrete Map	0x01	Map Pointer	Discrete check points, laps -> {0x0, 0x73, 0xe6, 0x153}	16bit	0x208	1		A few (unsure why)			Discrete Map	0x129b38
Character info	0x02					1		~100			Object	0x192098
Hidden Counter	0x02	Scarab Counter	total destroyed in all rounds, 0 to 300	16bit	0x234		100				???	0x19ded0
Hidden Counter	0x02	Scarab Counter	total destroyed in rounds 1+2, resets for round 3 {0 > 200 > 0 > 100}	16bit	0x234		101				???	0x1a1118
Hidden Counter	0x02	Scarab Counter	total destroyed in round 1, resets for rounds 2+3 {0 > 100 > 0 > 200}	16bit	0x234		102				Hidden Counter	0x1b2868
Character info	0x03					0		~300?			Monkey	0x1ccdb8
Character info	0x05					~3		0			Phase	0x1e87e8
???	0x05	Enemy Map Pointer x2	starts at 0 ish, ends at 1	Float	???	???		???			???	0x1edd40
Character info	0x07					0		<60			???	0x1f7378
Object	0x07	Door	>0 when closed, <0 when open	Float	0xc		0		0		Gem	0x209a78
Papaya	0x09	Papaya in hand	0 if not held, 1 if held	16bit	0x2f8	0		~150			Coin/Money	0x209a78
Papaya	0x09	Papaya not under tree	0 if moved from its spawn, 1 if it's where it spawns	16bit	0x74	0		~150			Scarab	0x20f770
Character info	0x09					~10		~150			???	0x21f2b0
Character info	0x0a					21		~30			Hidden Timer	0x24c990
Monkey	0x0a	Monkey in Hand	0 if not held, 1 if held	16bit	0xa54	0-20		~30			Target / Ring	0x250200
Monkey	0x0a	Monkey not Caught	0 if not in cage, 1 if in cage	16bit	0xa58	0-20		~30			Papaya / Boulder	0x276cf8
Character info	0x0b					4		<100			Projectile	0x348b98
Phase	0x0b	Boss Phase	0 before starting, odd # minions, even # vunerable, 7 boss fight over	16bit	0x3fc		20		?		Breakable Door ?	0x34e888
Phase	0x0b	Boss Minion Counter	starts at 3 every odd phase, moves to even phase when hitting 0	16bit	0x400		20		?			
Phase	0x0b	???	Only used to help differentiate the above node	???	???		21		?			
Character info	0x0c					0		0				
Character info	0x11					~30?						
Character info	0x17					0		0				
Character info	0x1b					0-4		1 (Angelica)				
Character info	0x1e					0		<25				
Character info	0x20					0		A lot?				
Hidden Timer	0x20	Timer	Counts up from 0 to 60, game ends at 60	Float	0x9b0	1		A lot?				
Character info	0x24					0		<50?				

Character Info Node Information			
Data	Code Notes	Size	Final Offset
X Position	These position/rotation memory slots also hold true for other types of nodes	Float	0x0
Z Position		Float	0x4
Y Position		Float	0x8
Rotation	Around some axis, only used once for a door opening in Baby Marv	Float	0xc
Ring Counter	Special case of item counter for minigame Ring Roller Coaster	16-bit	0x1f4
Camera Mode	1 third person, 0 first person, might only work when not on a mount	8-bit	0x258
Not Cutscene Mode	1 if there are no black bars on top and bottom of screne, 0 if there are	8-bit	0x26c
X Camera Angle	Next three while in first person, coordinates of the unit sphere vector for direction of camera	Float	0x5e0
Z Camera Angle	Locked between -0.6 and 0.7	Float	0x5e4
Y Camera Angle		Float	0x5e8
Health	1 is full, 0 is empty, % full	Float	0x7b4
Floor	What floor of the hub world a baby is on	16-bit	0x7e8
Item Counter 1	When there is an onscreen counter for an item	16-bit	0x8d4
Item Counter 2	When there is a second item counter on screen	16-bit	0x8d6
Shoes Spell	Seconds left of spell, starts at 2.0 in tough	Float	0x928
Shield Spell	Seconds left of spell, starts at 20.0 in tough	Float	0x92c
Potion Spell	Seconds left of spell, starts at 20.0 in tough	Float	0x930
Checkpoints	Checkpoint location, used in Papaya, starts at 0	16-bit	0xaf4
Magic Ball	Magic helper ball location, for Acrobatty Dash and Holy Pail ends at 0x28 and 0x21 repec.	16-bit	0xaf8
Continuous Map Pointer	Goes from 0 to 1, % of way through map	Float	0xcc8
Special Timer	Timer for Acrobatty Dash in seconds, counts down to 0	Float	0xcfc

Starting pointer is held in memory at 0x50b944
Nodes listed from the start can be moved a node further in the list every time a new object spawns,
and a node from the end can be moved closer to the end every time a node is destroyed
Character info nodes seem to be unique other than in the hub level (0x1b) where there is one per baby

Going forward in the list will have an addaddress chain that goes:
Starting pointer > +0x10 > (+0x14) # of node times > +0x0 > +Final Offset
Going backward in the list will have an addaddress chain that goes:
Starting pointer > +0x44 > +0x30 > (+0x10) # of node times > +0x0 > +Final Offset

Going backwards in the list can sometimes break depending on the level, or I might be dumb
Very confusing, might need further research if other data that cannot be found is eventually needed
(Looking at you Gamecube port)

For instance, the logic to test if the snowcone counter
in level 0x1b changes from 0x20 to 0x1f at the start
of the level before nothing else spawns would look like

Node Type	0x111328	Size	16-bit
Nodes from start	4	Final Offset	0x8d4

		Mem	8-bit	Level	=	Value	0x1b	
We want to go from the start of the list,	AddAddress	Mem	32-bit	0x50b944				
so starting with the main pointer then	AddAddress	Mem	32-bit	0x10				
0x10, instead of 0x44 and 0x30	AddAddress	Mem	32-bit	0x14				
	AddAddress	Mem	32-bit	0x14				
Four 0x14's, since we want the 4th node	AddAddress	Mem	32-bit	0x14				
	Remember	Mem	32-bit	0x14				
	AddAddress	Recall						
	AddAddress	Mem	32-bit	0x0				
	AddAddress	Mem	32-bit	0xd8				
	AndNext	Mem	32-bit	0xec	=	Value	0x111328	
	AddAddress	Recall						
	AddAddress	Mem	32-bit	0x0				
	AndNext	Delta	16-bit	0x8d4	=	Value	0x20	
	AddAddress	Recall						
	AddAddress	Mem	32-bit	0x0				
		Mem	16-bit	0x8d4	=	Value	0x1f	

Level Check
Access the Pointer
Access the Node
Testing Node Type
Delta Check
Mem Check

AddAddressing the 0x0 after the remember
is so we can check multiple nodes in a row
See to the right for an example

Of course, this check wouldn't work since throwing a snowcone
spawns a new object, pushing the character node back
one node, meaning the delta check will break the frame the
value decreases. This is just an example.

You must be very careful with deltas to make sure they don't
line up with the node moving the exact frame you read them
Checking if the node type's delta was also the right number would
help fiter erroneous triggers, but still potentially miss real triggers

For checking several nodes in a row for the snowcone data,
the logic may look like this, with the purple chunk from
the example on the left collapsed into a single line

		Mem	8-bit	Level	=	Value	0x1b	
	AddAddress	Mem	32-bit	0x50b944				
	Remember	Mem	32-bit	0x10				
0th node	Purple chunk copied from left with addHits in the last line							
	AddAddress	Recall						
	Remember	Mem	32-bit	0x14				
1st node	Purple chunk copied from left with addHits in the last line							
	AddAddress	Recall						
	Remember	Mem	32-bit	0x14				
2nd node	Purple chunk copied from left with addHits in the last line							
	AddAddress	Recall						
	Remember	Mem	32-bit	0x14				
3rd node	Purple chunk copied from left with addHits in the last line							
	AddAddress	Recall						
	Remember	Mem	32-bit	0x14				
4th node	Purple chunk copied from left with addHits in the last line							
	AddAddress	Recall						
	Remember	Mem	32-bit	0x14				
5th node	Purple chunk copied from left with addHits in the last line							
		Value		0x0	=	Value	0x1	(1)

The addhits are neccessary to act as an orNext chain without needing to split
this logic into seperate alt groups, as the Remember / Recall is impoartant to keep
logic length down to fit in the 64k character limit

This logic in the code is done with the function

```
comparison(data.levelIDLoaded, '=', 0x1b)
data.chainLinkedListRange(0, 5, ARRAY, true)
"0=1.1."
```

The inputs of the middle line being

0 - Start at node 0
5 - End at node 5
ARRAY - an array with an element per check you want to do per node, in this case 3.
The three bit of logic to the left that are green, orange, and red
true - We want to go forward through the list, not backwards