# Harvest Moon: Friends of Mineral Town Link-Up Patch Codes FAQ

by Eggie                                                        Updated to v1.2 on Dec 20, 2003

```
----------------------------------------------------------------------------
HARVEST MOON: FRIENDS OF MINERAL TOWN - U.S. VERSION
Link-Up Patch Codes FAQ
Version 1.2
Compiled on November 29, 2003
Last modified on December 20, 2003
By Eggie (eggiex1 (at) yahoo.com)
----------------------------------------------------------------------------
```

```
--------------------------------------
LEGAL NOTICE:
--------------------------------------
```

```
--------------------------------------
CONTENTS:
--------------------------------------
```

```
--------------------------------------
Xi. VERSION HISTORY
--------------------------------------

Version 1.2
December 20, 2003

- Added stage 2 codes to sections 11 and 12
- Added information about stage 2 codes
    on section 1. Introduction
- Added section Xii. Todo List
- Added and removed some Q & A and Credits information
- Some minor grammar corrections

Version 1.1
December 13, 2003

- Added section 11. Data Sequence
- Added section 12. Potential Code Breaker Codes
- Added information about sections 11 and 12
    on section 1. Introduction
- Some minor grammar corrections

Version 1.0
November 29, 2003

- First version.

--------------------------------------
Xii. Todo List
--------------------------------------

I decided to add this section to inform all readers of everything that is
left to be found that is link-up related. Once all of the following details
are resolved, this FAQ may be labeled as final. If anyone can help out and
has some answers for me, let me know!

1. Transferred names:
The only thing I need to do here is find out exactly what is the maximum
length of the names allowed in AWL. As mentioned in section 9, the names must
be null terminated, and because of this its length cannot be determined via
a code. Since I don't have AWL, I have no way of knowing what is the maximum
allowed name length for each of the transferred names. Anyone who has AWL
and can tell me this please e-mail me the information and I will give you
credit in a future update. I would need to know the max length of each of the
3 names separately in case there is the remote chance they might differ. The
names needed are, of course, the player, the farm, and the player's child.
This means that whoever would provide me with this information has to at
least get a child!

2. Mysterious book left:
The book shelf on the library from section 10 holds 36 books with information
about 36 different characters from AWL. Among these names is the player name.
However, there is a 37th book that is supposed to be transferred over from
AWL, which contains information about your child from AWL. The 37th book is
not on that shelf. It must be somewhere else. I have predicted that the slot
for that book is right after the player info as described in section 10.
Furthermore, the codes for the books make this assumption as well. So anyone
who uses the codes from section 12 has a chance to come across this book.
```

All I need to know is where that book is. This way, I can verify that the child's book info is indeed after the player's and so that I can also describe the location of that book.

3. The Doctor's reference to AWL:
This I'm not too sure about. In section 8, I describe which bits correspond to each of 10 different characters who make a reference to AWL. However, the Doctor is supposed to have a reference about it too. I assumed that it might not be link-up related and that his reference may be accessed without a code. However, I'm not entirely too sure about this. I do know that he makes no reference to any of the transferred names, so it is possible that he might make this reference without linking up. I need someone to let me know what event will cause him to make a reference to Forget-Me-Not Valley so that I can test this. If his reference is link-up related, however, then the only way to activate it is to make a change to the code I posted. To test this, use the codes 32004799 00FF and 3200479A 007F. Even with those codes, though, he will still not make the reference, which is what led me to believe it might not be link-up related, but it is possible that some other event must be triggered first as well.

As far as I know, those are the only 3 things which are still undetermined. If anyone knows of anything else that's link-up related and that has not been addressed, or if anyone can help me on these 3 remaining details, please do. As always, I'll be sure to give you credit for the finding. :)

--------------------------------------
## 1. INTRODUCTION
--------------------------------------

-------------------
### A. About stage 2 codes
-------------------

WARNING: Unless otherwise noted, all addresses and codes are on stage 1!

First off, read section 4 part C to find out what is meant by a different stage code. Now that you understand this, here are some details about stage 2 codes I added.

As far as all link-up codes posted here are concerned, the stage 2 codes are all exactly 10,292 bytes further down from the stage 1 codes. I say as far as the link-up codes goes because it is possible that this difference in offsets may differ for different data. However, I tested this with all addresses posted here just to be sure, including all non-required addresses. (section 3 describes what I mean by a required 'R' address)
10,292 in hexadecimal is 2834. So for ANY address posted on this FAQ that is on stage 1, add hexadecimal 2834 to it and the result is the stage 2 address. Of course, you must remember that you are adding in hexadecimal!

Now that I have verified this offset difference is the same for all addresses here, I do not need to write them all out myself. However, to make this FAQ more complete, I decided to add it anyways.

Sections 5 - 10 do not have any stage 2 information for them. Those sections are only going to be read by those who desire to make their own codes. People who make their own codes from that information will definitely know how to add 2834h to those addresses if they want the stage 2 codes.

Sections 11 and 12, on the other hand, have the stage 2 codes in them as well, since many people who will just go directly there will likely not know or not care to do the math. I describe the format of the codes in those sections.

One final note which is obvious but nonetheless I will mention, for anyone who still doesn't understand stage 1 and 2 or what stage they currently are in, the most simple thing to do is to try a stage 1 code, and if the stage 1 code isn't working, then try the stage 2 code.

```
-------------------
B. About sections 11 and 12 and e-mailing me
-------------------
```

Before you ever decide to e-mail me any questions, make sure you read the other sections as well. If you only read sections 5 - 10 and get some problems, chances are you are using codes whose addresses are also used for something else. (read part B to see an example) So in such case, section 11 would answer your question. If on the other hand, you only read sections 11 and 12, you won't have all the detailed information that you're probably missing. So the point is, if you don't want to read the whole thing you can still make everything work, but please don't e-mail me questions if you don't read the other sections, as it is likely that doing so will answer the questions for you.
If neither the stage 1 nor the stage 2 codes are working at all for you while using the exact codes I posted on section 12, then please don't hesitate to e-mail me about it. In that case, it is possible that I might have added something wrong. I made every effort to make sure I made all calculations correctly but with so many values it's possible I may have made a mistake.
It is also possible that there might be yet another stage, which would be equally an interesting discovery.

```
-------------------
C. About section 11. Data Sequence
-------------------
```

If you plan to make your own codes instead of using the ones from section 12, section 11 will be of great use to you. It clearly shows what addresses for one code collide with another. For instance. If you read section 6, you might want to use the code

```
3200479C 0008
```

to get Lou to visit you. Then, when you read section 7, you would want

```
3200479C 0010
```

to get Lu to visit you. Notice any problems? If you don't, don't even bother reading on. Just go to section 12 and cross your fingers those codes will work flawlessly for you. On the other hand, if you noticed those 2 codes have the same address, maybe you can use this FAQ after all.
Obviously, you would want to combine the 2 values and use

```
3200479C 0018
```

as a code. If you do, you would still be wrong! Turns out that the 3 low bits of that address are used for yet another thing; part of the counter of how many recipes Lu has for you. A value of 0018 as used above would mean she has no recipes for you at all. (or 1 if you set the high bit of the previous address)

This game's data is very tricky, as you can see. Section 11 shows the sequence of the data very clearly and will let you decide on what values you want to use without having to search the rest of the FAQ to detect address collisions. It sure helped in making section 12, at least.

-------------------
D. About section 12. Potential Code Breaker Codes
-------------------

For anyone who has already read sections 1 - 10 (or at least understands sections 2 and 4) feel free to go directly to section 12 and use codes from there. However, please note that any questions e-mailed to me regarding code-related side effects will be ignored. I added section 12 to help people who just want some codes in getting what they want without asking me over and over. Section 4 explains fairly well the impossibility of eradicating side effects altogether. The normal sections 5 - 10 contain all the information needed to work around these side effects. (although admittedly the explanations might need a revision since some people don't understand them)

In any case, I made sure section 12 would be somewhat flexible, so that with a little thought from your part, you might be able to avoid them. That said, please do not be offended if I ignore your e-mail for asking something that can be easily understood from this FAQ.

-------------------
E. About
-------------------

For those who have no idea what this FAQ is all about, it is for the link-up codes for the game Harvest Moon: Friends of Mineral Town. When this game is connected to the Game Cube game Harvest Moon: A Wonderful Life (AWL), there are specific bonuses that may be unlocked in this game and in the Game Cube version as well. However, for those who don’t have the Game Cube version or won’t be getting it at all, they will normally never be able to experience these bonuses that are acquired only by link-up to the Game Cube’s version... until now.

This FAQ will describe these bonuses, where they are represented in the game’s data segment, and for those who have a cheat device, how to create codes that would allow them to receive these bonuses without ever linking up to the Game Cube.

However, it should be noted that upon normal link-up to AWL, much data may be transferred over that may not all be in this FAQ. As I find out about more and more of these bonuses, I will be updating this FAQ accordingly.

I found these addresses without linking to the Game Cube version at all (at the time of this writing it wasn’t even released!) so it is possible that there may be side effects as a result (although I have not yet found any with the current set of addresses). In any case, you should be aware at all times that creating any codes from this data (for your own personal use only of course) and applying them to your game is a conscious decision of yours and any problems that may be caused are not my responsibility in any way. Especially if you create invalid codes because you didn’t understand this FAQ too well! Read section 4 to understand why codes you may be making from this FAQ are causing you side effects.

--------------------------------------
## 2. BASIC TERMINOLOGY, ACRONYMS AND RELEVANT INFORMATION
--------------------------------------

This section may be skipped by anyone who has a profound (or at least somewhat concrete) knowledge of data representation (as far as a code is concerned).

```
-------------------
A. Bits and bytes
-------------------
```

A bit is simply a single 0 or 1 value. 8 of these compose a byte. Memory is almost always arranged into pieces of 8 bits (i.e. a whole byte) Think of a byte as an atom and the bits as its sub-atomic particles. Sub-atomic particles by themselves cannot really do anything, and they must be a whole atom to be of any use. Likewise, a whole byte is just about as small as you can get.

Conventionally, bits are stored from least to most significant. This is the reverse of how we usually write numbers. This type of convention is called Little-Endean order.

This game, (along with almost every other game I've played) uses this method to store data. Therefore, bit 0 is the least significant bit and bit 7 is the most significant bit in a byte.

```
-------------------
B. Memory addressing
-------------------
```

A memory address is simply a location in memory where some data is stored. At least as far as this game is concerned, a memory address is composed of 7 digits in hexadecimal. The value at that address is 2 digits in hexadecimal that is currently at that location.

To avoid getting into too much detail, I will quickly explain what is meant by an 8 bit, a 16 bit and a 32 bit value at a particular address with the following example:

Suppose we have the following memory address:

02004070

This points to one specific byte, which is 8 bits. If we use an 8 bit value at that address, then only the value at that address will be replaced.

Now suppose we use that same address and want to use a 16 bit, or 2 byte value to insert there. Obviously 2 bytes won't fit into 1 byte, so what happens is, the 8 least significant bits will occupy 02004070 (the address we specified) and the other 8 bits will occupy 02004071 (which is the address that follows it)

The same happens with 32 bit addresses. So if at any point you see an address whose value is to be replaced (or has a current value) that is more than 8 bits, then, from least significant to most significant, the bits will be placed in every other address until no more bits are left.

So for the last relevant type of memory value we could use, if we placed a 32 bit value into the same address of 02004070, then the least significant 8 bits will go into 02004070, the next least significant go into 02004071, the next least significant go into 02004072 and the most significant go into 02004073.

This is important to know because at the bit level, if one particular bit is taking the highest order of an address and an additional bit or two is needed by the game, then the lower bits of the next address will be used to store that information. This happens a lot in this game, as we should soon examine.

-------------------
## C. Flags
-------------------

Almost every game needs to represent true and false statements at some time. For instance, in this game, there is a need to represent whether or not Won has come by and visited you already for the first time. If he has, he will not visit you again (not “for the first time”).

Obviously, this can be done with a single bit. Conventionally, a value of 0 indicates false (or that an event has not taken place) and a value of 1 indicates true (or that the event has taken place). But the smallest component is a byte, and we cannot take a single bit and use it unless it’s within a whole byte.

This is where flags come in. A flag is a byte whose bits are used to represent true and false statements. Since each byte has 8 bits, 8 statements, or in a game 8 events, can be represented with 1 single byte. Obviously, this is much more efficient than the alternative; using a whole byte to represent a true or false (Boolean) statement!

As you may have guessed by now, flags are used in this game to represent events. Therefore, to represent, for example, whether or not Lu has been transferred over and should visit you, a bit in a flag may be used where all other bits are used to represent other events as well.

--------------------------------------
## 3. PERSONAL FORMATTING
--------------------------------------

This section should be read by everyone because it describes the conventions I will be using in the following sections.

As mentioned before, the standard is to refer to the least significant bit in a byte as bit 0 and the most significant as bit 7. However, I call the least significant bit 1 and the most significant bit 8.

The way I write an address whose bits cause different behavior (i.e. flags) is usually as following:

```
xxxxxxx = (Overall description of that this address does)
1 = (Description of what this bit does)
2 = (Description of what this bit does)
3 = (Description of what this bit does)
4 = (Description of what this bit does)
5 = (Description of what this bit does)
6 = (Description of what this bit does)
7 = (Description of what this bit does)
8 = (Description of what this bit does)
Where xxxxxxx is the address.
```

In the case that I only have information on a single bit, I will write it as this:

xxxxxxx : n = (Description of what bit n does)
Where xxxxxxx is the address and n is the bit, a number between 1 and 8.

In the case that there is a range of bits within a byte that count something, I will write it as this:

xxxxxxx : m-n (Description of what this bit range does)
Where xxxxxxx is the address, m is the lower (least significant) bit and n is the higher (most significant) bit of the specific range of bits.

I will now explain about how I split a counter byte into its bits. Suppose we have an address xxxxxxx and that address counts the amount of money you have. The following results are caused by setting each individual bit in that address:

| Set bit when it was cleared | Effect on money by setting this bit |
|---|---|
| 1 | Money increases by 1 |
| 2 | Money increases by 2 |
| 3 | Money increases by 4 |
| 4 | Money increases by 8 |
| 5 | Money increases by 16 |
| 6 | Money increases by 32 |
| 7 | Money increases by 64 |
| 8 | Money increases by 128 |

For those of you who know binary fairly well, you can see why this is the case. As we go into more and more significant bits, setting that bit would yield a greater result. So if I do this:

xxxxxxx = Money x 256

It means that an increment of 1 in that address would increment the money by 256 (i.e. this would be the second byte of the money)

This information is important because sometimes, you might see something like this:

xxxxxxx : 1-3 = Money x 32

What that means is that an increment in the range of bits 1 - 3 would cause an increment by 32 to the money. Obviously, the other bits that would cause an increment by 1, 2, 4, 8 and 16 are on the previous address, and are the last (most significant) 5 bits of the previous address.

One last issue, you will notice that all the xxxxxxx I wrote are 7 characters long. As I mentioned in the previous section, addresses in this game can be represented with only 7 characters. The character missing to complete a code is the one that would describe the code type. For instance, 3xxxxxxx is an 8 bit code for Code Breaker. I neglected the leading digit on all the addresses because they are up to the user to decide on what value to use, dependent on their cheat device.

All addresses contained here are all the addresses that are related to the link-up bonuses. However to acquire all these bonuses, not all codes must be entered. For instance, to get Lu to visit you, you must set a bit at a specific address. For Lu to appear at the Inn there is another address I also posted but that it’s not needed. By using only the address for Lu to visit you, after she visits she will appear at the inn and no additional code will be needed. However, since her appearance at the inn is link-up related, I also have that address here.

Any address that is required to unlock everything will be written as follows:

```
R:
xxxxxxx = (Code information here)
```

Any code that does not have the R heading above is not required, and is simply an event that can be set by the player once all the required codes have been used. So for the best possible game, the non-required codes shouldn't be used because of the side effects issue. (see section 4) However, they may spark some interest to anyone that wants to play around with those.

Required codes are not really "required." However, if the player hopes to unlock all the bonuses (discovered so far in case these are not all of them) then those are all the relevant addresses that must be tinkered with. Neglecting to use a required code for a specific bonus simply means the player won't get that specific bonus.

This is obvious but I thought I should mention it because by using the term "required" it might lead others to believe they must use all those codes for just one single bonus they would want. So I thought I would clarify this by saying that, by "required", I mean it is required if the player hopes to get all the bonuses.

--------------------------------------
## 4. PROBLEMS WITH CODE CREATING IN THIS GAME
--------------------------------------

As far as I know, there are 3 problems with creating absolute codes in this game, and hence the purpose of this FAQ:

-------------------
### A. What to do about flags
-------------------

Suppose we know that at address xxxxxxx, the 5th bit determines whether or not Lu has been transferred over to your game. Now we know we want a code that would set bit 5. The value we would want at this address is therefore 00010000 or 10 in hexadecimal. But what about the other bits? If we use 10 as the value, we would be forcing all the other bits to be 0. This would be acceptable if they were all cleared to begin with, but what if they were not?

Suppose bit 2 represents whether or not the Goddess has visited you for the 10,000 step accomplishment. Then we would be clearing that event altogether!

To come up with the ideal value to use, we would have to know what bits are currently set on each address we want to alter. Cheat devices, unfortunately, do not tell the user of the currently stored value. Unless we know what all the bits in the byte do, we cannot come up with the ideal value to use unless we guess our way to one. As a result, using a value that will set or clear other bits without knowing about it will lead to SIDE EFFECTS. For this reason, I am hesitant to create absolute codes, and I'm leaving the readers of this FAQ to decide what values best fits them for their game.

```
-------------------
B. Alignment issues
-------------------

A segment of data is said to be byte-aligned if, for every byte in the
range other than flags, every bit in that byte represents ONLY one
particular object.

For instance, if we have 02004070 represent the low byte of the money
value, then that byte is byte-aligned. Suppose, however, that instead
of using bits 1 - 8 for the money, we have bit 1 to represent whether
or not you have spoken to Won, and then bits 2 - 8 and bit 1 of 02004071
represent the money. Then in this case, 02004070 is not byte aligned,
because it is not a flag (since the 7 highest bits are counting money
and are therefore not true/false bits) and bit 1 is used for something
other than counting money.

In this game, unfortunately, there are a lot of data segments that are
not byte aligned. For the same reasons as that of the flags, making codes
that will set specific bits only but leave the others alone is not
currently possible. For our example above, we could not use a code that
would set bits 2 - 8 and leave bit 1 as either set or cleared depending
on what it is. The user must either use a value that will set it or clear
it, and if such user does not know what it's for, this decision may prove
to be difficult.

-------------------
C. Jumping data
-------------------

Unfortunately, data in this game is not constant. For one game session,
the value for money may be at address 02004070 but for another play of
the same game on the same cartridge it may be on 02006258!

As far as it is currently known, however, there are only 2 different
possible addresses. These have been referred to as Stage 1 and Stage 2.

Stage 1 appears to be the ordering of the data when a new game is started,
and before the game is saved and reset. Stage 2 is after this fact.

Obviously, a game will be on Stage 2 for most of its time. Unless otherwise
noted, all codes here are on stage 1. Specifically, all addresses in
sections 5 - 10 are in stage 1 only, while those found in sections 11 and 12
are in both. The format of those addresses is described on those sections.

-------------------
D. Future solutions for A and B
-------------------

There is one way to solve the problem of setting a bit without affecting
the others: The logical OR operator.

The logical OR operator used on two bits has the following truth table:

A value     B value     Result of OR
   0           0            0 (= the value of B when ORed with a 0)
   0           1            1 (= the value of B when ORed with a 0)
   1           0            1 (= set to 1 when ORed with a 1)
   1           1            1 (= set to 1 when ORed with a 1)
```

As we can see, if we have a bit and we OR that bit with 1, we set that bit.
If we OR the bit with a 0, we leave it as it is, whether it is set or not.

For instance, if we want to set bit 3 at 02004070, we would OR the value
of that address with 00000100, or 04. This would be a hypothetical result:

```
Value at memory:   01001001
Value to OR with:  00000100
Result:            01001101
```

If we know what bit we want to set, we can use OR to set only that bit and
preserve all other bits. A code that would OR the value of an address with
a specified value would accomplish this, and wipe out all code-related
side effects!

Unfortunately, there is no such code type at this time that I know of.
Code Breaker has a logical AND code type which is used to clear a bit,
but we want to set bits, not clear them. If there is ever a cheat device
made that would accomplish this, then you know what to do!

--------------------------------------
## 5. LINK UP STARS AND THE SEASIDE COTTAGE
--------------------------------------

In this game, there are stars that can be acquired by linking up to the
Game Cube. These stars lead to the many different bonuses that can be
acquired only through link-up.

Normally, when a connection is made, stars are awarded to the player in
this game. At that point, the game will count how many stars have been
acquired and based on that it will give the player some bonuses.

Unfortunately, the execution of the code that would give the player the
bonuses is executed at some point while connected. Therefore, the bits
that would give the user Lu, Lou, and other bonuses will not occur with
just a stars code! However, the Seaside Cottage is awarded to the user
by the Goddess after the link is over, and therefore this code alone will
award at least that additional bonus to the player.

The link up stars addresses are as follows:

```
R:
200479E : 2-8 = Link-up star points x 1
200479F : 1 = Link-up star points x 128
```

The Star points required for each star are as follows:

| Stars | Points |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 5 |
| 3 | 9 |
| 4 | 13 |
| 5 | 17 |
| 6 | 21 |
| 7 | 25 |
| 8 | 29 |
| 9 | 33 |
| 10 | 37 |
| 11 | 41 |

| | |
|---|---|
| 12 | 45 |
| 13 | 49 |
| 14 | 53 |
| 15 | 57 |
| 16 | 61 |
| 17 | 65 |
| 18 | 69 |
| 19 | 73 |
| 20 | 77 |
| 21 | 81 |
| 22 | 84 |
| 23 | 87 |
| 24 | 90 |
| 25 | 93 |
| 26 | 96 |
| 27 | 99 |
| 28 | 102 |
| 29 | 105 |
| 30 | 108 |
| 31 | 111 |
| 32 | 114 |
| 33 | 117 |
| 34 | 120 |
| 35 | 123 |
| 36 | 126 |
| 37 | 128 |
| 38 | 130 |
| 39 | 132 |
| 40 | 134 |
| 41 | 136 |
| 42 | 138 |
| Err | 139 |

139 points or more will cause an “error”. It won’t show the stars and will cause the Goddess to skip the part where she gives you the Seaside Cottage.

At this point, I do not know for sure if bit 1 of 200479E is actually used for star points. If it is, then all these points are actually double their value and the addresses’ point weights are double too. Since there is no observable effect caused by setting bit 1 of 200479E only (note: NOT the same thing as adding 1 to the whole byte) then this cannot be determined. However, if that was the case, then there would be 2 possible values that would be considered by the goddess as “maximum connectivity” and this seems very unlikely, so there is a very high chance this information is correct as it is. Even so, if we assume bit 1 is used for something else (so that we do not tinker with it) and we give ourselves 138 points based on the table above, we will get maximum connectivity, the seaside cottage, and prevent any possible side effects in case bit 1 is used for something else after all.

If this code is used right before she is about to show you your stars, then she will say something about how many new things will happen. If used before, she won’t say this. In either case, you will still get the Seaside Cottage after she shows you your stars, and either way, the bits that give you the other awards are not set. As I mentioned before, this is an indication that the code that would set those bits occurs at link up, and so other additional codes must be used to acquire these special characters and bonuses.

Since the Seaside cottage itself is link-up related, this is its address:

20025D8 : 3 = Seaside cottage obtained

This address has been discovered by other users before, because the whole address itself deals with the bonus houses the player can acquire. I found this independently, but it is only fair to mention other users have discovered it as well. The link up stars address, however, was never posted before.

--------------------------------------
## 6. LOU: ARTIFACTS COLLECTOR
--------------------------------------

Lou (a.k.a. Van) is the first character that's supposed to be transferred over to this game from AWL. He is supposed to visit you on the first Wednesday after you have received him from link up.

The following address will let Lou come visit you on the next Wednesday:

```
R:
200479C : 4 = Lou visits
```

If on such Wednesday it is raining, or there is a storm or something, he won't come until next Wednesday. When he does appear, he will greet you at the door as soon as you step outside that morning.

Once he has met you, he will be at the Inn's second floor every Wednesday. (including that same day) After he has visited you, the following bit will be set:

```
200475E : 7 = Lou is in the Inn (he already visited you)
```

This code does not need to be entered, however, since the game will take care of it normally after he greets you at the door. A code to set that bit will cause him to have visited you already, and he will just be at the inn instead.

Once you go to visit him at the Inn, you can buy items he has for sale. The first important item he sells is the record player. The following bit is set when you buy it:

```
20027CD : 4 = Bought record player
```

This code is also not needed if you intend to fork over the $2000 for the record player.

Lou is also supposed to sell records after more link-ups are made to the Game Cube. The following address takes care of that:

```
R:
20045D2 = Records sold
```

This address is actually byte aligned. (thank goodness) It indicates what record number he is up to. For instance, a value of 7 means he is currently selling all records up to record 7. Since he only has 10 records to sell, the value to use here should be 10, or 0A in hexadecimal.

Once the record player and records have been purchased, you can put a different record on the machine. The following address determines what record is on the record player:

```
2002954 : 1 = Record on record player (must be set)
2002954 : 2-5 = Record type

bits 2-5 composite value:
0 = Album 1
1 = Album 2
2 = Album 3
3 = Album 4
4 = Album 5
5 = Album 6
6 = Album 7
7 = Album 8
8 = Album 9
9 = Album 10
```

So if you want Album 6 on the record player, bits 2-5 must equal 5, and bit 1 must be set so this is
0000 101 1 = 0B in hexadecimal.

---------------------------------------
## 7. LU: THE RECIPES GIRL
---------------------------------------

Lu (a.k.a. Luu) is supposed to be the second character that crosses over to this game. She is supposed to visit you the first Sunday after you have earned her through link-up. Just like Lou, she will greet you at the door that day as soon as you step out, so long as the weather is good, but unlike Lou, she won't come on Winter at all! There might be some other seasons or conditions in which she won't come, but as long as she crossed over, she will visit you the first Sunday that she can.

In order for Lu to visit you, the following address is needed:

```
R:
200479C : 5 = Lu visits
```

Just like Lou, if she has visited you already, the following bit is set:

```
200475F : 5 = Lu is in the Inn
```

Lu is supposed to give you recipes. She has at most 10 available, and she will give you 1 recipe per Sunday. The following bit is set when she has already given you a recipe that Sunday:

```
20047B4 : 2 = Lu gave you a recipe this week
```

If set to 0 again, she will give you another one! However, she must have recipes available for you as well. The following addresses determine how many recipes she has to give you:

```
R:
200479B : 4-7 = Times Lu will tell you a recipe
200479B : 8 = Recipes Lu has left x 1
200479C : 1-3 = Recipes Lu has left x 2
```

Both of those data segments should be set to a value of 10, since she only has 10 recipes. This means that bits 4-7 of 200479B should be 0, 1, 0, and 1 respectively, bit 8 should be 0, and bits 1-3 of 200479C should be 1, 0 and 1 respectively. (remember this is little-endian order)

Keep in mind, however, that if you do not have a kitchen yet, then no matter what you do she will not give you any recipes. So if you do not have a kitchen that is the reason why these codes are not working.

As Lu gives you more and more recipes, a counter must be kept to keep the game informed of how many recipes you have been given, so that the next recipe you receive is not just chosen at random. The following addresses take care of this:

```
200479A : 8 = Recipes Lu has told you x 1
200479B : 1-3 = Recipes Lu has told you x 2
```

This value should be left alone since it is initially 0, but can be used to change which recipe is to be acquired next. The recipes she gives you are in the following order:

```
Recipe 1 = Wild Grape Juice
Recipe 2 = Corn Flakes
Recipe 3 = Buckwheat Chips
Recipe 4 = Mountain Stew
Recipe 5 = Toasted Rice Cakes
Recipe 6 = Baked Corn
Recipe 7 = (antidote for Fatigue)
Recipe 8 = Vegetable Juice
Recipe 9 = Curried Bread
Recipe 10 = Apple Soufle
```

---------------------------------------
8. FORGET-ME-NOT VALLEY VILLAGERS’ REFERENCES*
---------------------------------------

*None of the addresses here need to be altered in order to acquire all the bonuses. The reason is that characters are supposed to make a reference to AWL only once, and then they will never talk about it again. There is therefore nothing permanent added to the game by altering these addresses.

After link-up to the Game Cube, characters from this game are supposed to make references to your character and your farm in AWL. The following bits, if set, will cause these characters to say something about AWL:

```
2004799 = Forget-Me-Not Valley References
        1 =
        2 =
        3 =
        4 = Lou
        5 = Lu
        6 = Ellen
        7 = Barley
        8 = Rick

200479A = Forget-Me-Not Valley References
        1 = Thomas
        2 = Jeff
        3 = Carter
        4 = Gotz
        5 = Sprites
        6 =
        7 =
        8 = (Used for something else)
```

Bit 8 of 200479A is used for Recipes Lu has told you x 1, so there cannot be any more bits or bytes used for this after the 7th bit. Although I have no information on bits 6 and 7 at this point, this does not necessarily mean they are used for references. They may be used for something else.

Notice that there are no gaps in the sequence of bits used for references. This implies that all characters that reference AWL are only these. There are also 10 characters only, which is also a magic number (like 10 Albums from Lou and 10 recipes from Lu). There is supposed to be an additional reference to AWL by the Doctor, but there is a chance that it is not link-up related. He mentions something about the other doctor in AWL.

--------------------------------------
9. TRANSFERRED NAMES
--------------------------------------

When characters make references to your character from AWL, and when you read the books on the 2nd floor of the library (more on that in the next section) your character's name, your farm's name, and your child's name in AWL are mentioned in this game. Therefore, these 3 names are also transferred over. The following addresses are used for those names:

```
R:
20047B8 - ????? = Transfer Name 1 (Farm)
20047C8 - ????? = Transfer Name 2 (You)
20047D8 - ????? = Transfer Name 3 (Child)
```

You will notice I put "?????" for the upper limit. At this point, I do not know how many characters long the name is supposed to be.
The last character in the name must be a 0 (null termination) and therefore until it reads a 0 the game will just continue reading all subsequent addresses as part of the same name. To solve this, we only need to know the intended limit of the name, change the upper limit and set the character that follows it to 0. As soon as I know what that upper limit is for sure, I will adjust this. All I need to find this information is for someone who plays AWL to tell me how many characters are allowed for each of those names. Someone that has played a different version of AWL from a different region said it was 8, although it might be more or less for the U.S. version. We will have to wait until the U.S. version is released.
It is obvious by the above addresses that the name is not intended to be more than 15 characters in length, since otherwise it would collide with the other names (15 characters + null character = 16 addresses)
The names are stored in standard ASCII format.

--------------------------------------
10. THE LIBRARY: SECOND FLOOR'S MYSTERIOUS BOOK SHELF
--------------------------------------

For anyone that has been on the second floor of the library, you will notice that the book at the center of the top shelf is empty. This shelf is supposed to be used for transfer information from AWL characters.
When one of such character's information is transferred, a certain amount of data is passed over. A number between 0 and 3 will tell the game how much information about that character is available in the book. Therefore, this data is not stored as flags, but as 2 bit counters.

The following are the relevant addresses for this:

R:

```
2004783 = Mary's Library.2nd Floor.Center Shelf Books
        1 =
        2 =
        3 =
        4 =
        5 = Player = (player name) part x 1
        6 = Player = (player name) part x 2
        7 =
        8 =

R:
2004784 = Mary's Library.2nd Floor.Center Shelf Books
        1 = Tak = Takakura part x 1
        2 = Tak = Takakura part x 2
        3 = Manna = Romana part x 1
        4 = Manna = Romana part x 2
        5 = Lumin = Lumina part x 1
        6 = Lumin = Lumina part x 2
        7 = Sebastian = Sebastian part x 1
        8 = Sebastian = Sebastian part x 2

R:
2004785 = Mary's Library.2nd Floor.Center Shelf Books
        1 = Wally = Wally part x 1
        2 = Wally = Wally part x 2
        3 = Chris = Chris part x 1
        4 = Chris = Chris part x 2
        5 = Hugh = Hugh part x 1
        6 = Hugh = Hugh part x 2
        7 = Grant = Grant part x 1
        8 = Grant = Grant part x 2

R:
2004786 = Mary's Library.2nd Floor.Center Shelf Books
        1 = Sam = Samantha part x 1
        2 = Sam = Samantha part x 2
        3 = Kate = Kate part x 1
        4 = Kate = Kate part x 2
        5 = Galen = Galen part x 1
        6 = Galen = Galen part x 2
        7 = Nina = Nina part x 1
        8 = Nina = Nina part x 2

R:
2004787 = Mary's Library.2nd Floor.Center Shelf Books
        1 = Daryl = Daryl part x 1
        2 = Daryl = Daryl part x 2
        3 = Gustafa = Gustafa part x 1
        4 = Gustafa = Gustafa part x 2
        5 = Cody = Cody part x 1
        6 = Cody = Cody part x 2
        7 = Kassey = Kassey part x 1
        8 = Kassey = Kassey part x 2

R:
2004788 = Mary's Library.2nd Floor.Center Shelf Books
        1 = Patrick = Patrick part x 1
        2 = Patrick = Patrick part x 2
        3 = Murray = Murray part x 1
        4 = Murray = Murray part x 2
        5 = Tim = Tim part x 1
        6 = Tim = Tim part x 2
        7 = Ruby = Ruby part x 1
        8 = Ruby = Ruby part x 2

R:
2004789 = Mary's Library.2nd Floor.Center Shelf Books
        1 = Nancy = Nami part x 1
        2 = Nancy = Nami part x 2
        3 = Rock = Rock part x 1
        4 = Rock = Rock part x 2
        5 = Garcia = Griffin part x 1
        6 = Garcia = Griffin part x 2
        7 = Muffy = Muffy part x 1
        8 = Muffy = Muffy part x 2

R:
200478A = Mary's Library.2nd Floor.Center Shelf Books
        1 = Carter = Carter part x 1
        2 = Carter = Carter part x 2
        3 = Flora = Flora part x 1
        4 = Flora = Flora part x 2
        5 = Vesta = Vesta part x 1
        6 = Vesta = Vesta part x 2
        7 = Marlin = Marlin part x 1
        8 = Marlin = Marlin part x 2

R:
200478B = Mary's Library.2nd Floor.Center Shelf Books
        1 = Celia = Celia part x 1
        2 = Celia = Celia part x 2
        3 = Hardy = Hardy part x 1
        4 = Hardy = Hardy part x 2
        5 = Van = Van part x 1
        6 = Van = Van part x 2
        7 = Mooky = Mooky part x 1
        8 = Mooky = Mooky part x 2

R:
200478C = Mary's Library.2nd Floor.Center Shelf Books
        1 = David = Nak part x 1
        2 = David = Nak part x 2
        3 = Nic = Nic part x 1
        4 = Nic = Nic part x 2
        5 = Brad = Flak part x 1
        6 = Brad = Flak part x 2
        7 =
        8 =
```

Some characters do not have 3-part information, and therefore you would get the same information with a value of 1 or 2 as you would with 3. However, a value of 3 should nevertheless lead to no side effects.

I have given 2 names for each character. As an example, we have:

```
1 = David = Nak part x 1
```

The first name, David, is the one that appears on the list of names to choose from on the books. But Nak is the name that appears when you select that book. So I’m guessing that’s just because some characters have aliases.

In any case, the first name is the list name and the second is the name that shows up on that character’s information when that character is selected.

You may have noticed a gap in the data too. After player, there is enough space for exactly one more character. I believe this is for the character’s son, since there is one more additional book information that’s supposed to be available for him. However, it does not show up on that shelf, so I have yet to encounter that book.

```
--------------------------------------
11. DATA SEQUENCE
--------------------------------------
```

I added this section because it shows the sequence of the link-up related addresses and what they do. Basically, this is a "map" of the data. Of course, a map without a compass is not very useful, so don't expect a full explanation of the addresses here. More thorough information can be found on sections 5 - 10. This section simply contains the information from said sections without an explanation.

Remember, an 'R' above an address indicates that the address must be altered if you hope to get all link-up bonuses. If at least 1 bit in that address needs to be set to satisfy this requirement, I added the 'R' above that address.

In this memory map, the bits without a description are the ones for which I have no information, because those bits are the ones used for non link-up related information. Every such bit is one possible side effect that a code may produce. Those addresses that have no "bit map" information are the ones which are byte-aligned. (the records sold and the transferred names only) which are also the only ones that will cause no side effects. The others may not cause any side effects if a correct value is chosen for them.

Stage 1 addresses are written as "1. xxxxxxx" and stage 2 addresses are written as "2. xxxxxxx". Remember, ONLY ONE of the 2 stages is needed, NOT both!

```
1. 20025D8
2. 2004E0C
        1 =
        2 =
        3 = Seaside cottage obtained
        4 =
        5 =
        6 =
        7 =
        8 =

1. 20027CD
2. 2005001
        1 =
        2 =
        3 =
        4 = Bought record player
        5 =
        6 =
        7 =
        8 =
1. 2002954 = Record on record player information
2. 2005188
        1 = Record on record player (must be set)
        2 = Record type
        3 = Record type
        4 = Record type
        5 = Record type
        6 =
        7 =
        8 =

R:
1. 20045D2 = Records sold
2. 2006E06

1. 200475E
2. 2006F92
        1 =
        2 =
        3 =
        4 =
        5 =
        6 =
        7 = Lou is in the Inn (he already visited you)
        8 =

1. 200475F
2. 2006F93
        1 =
        2 =
        3 =
        4 =
        5 = Lu is in the Inn
        6 =
        7 =
        8 =

R:
1. 2004783 = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FB7
        1 =
        2 =
        3 =
        4 =
        5 = Player = (player name) part x 1
        6 = Player = (player name) part x 2
        7 =
        8 =

R:
1. 2004784 = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FB8
        1 = Tak = Takakura part x 1
        2 = Tak = Takakura part x 2
        3 = Manna = Romana part x 1
        4 = Manna = Romana part x 2
        5 = Lumin = Lumina part x 1
        6 = Lumin = Lumina part x 2
        7 = Sebastian = Sebastian part x 1
        8 = Sebastian = Sebastian part x 2
```

```
R:
1. 2004785 = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FB9
        1 = Wally = Wally part x 1
        2 = Wally = Wally part x 2
        3 = Chris = Chris part x 1
        4 = Chris = Chris part x 2
        5 = Hugh = Hugh part x 1
        6 = Hugh = Hugh part x 2
        7 = Grant = Grant part x 1
        8 = Grant = Grant part x 2

R:
1. 2004786 = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FBA
        1 = Sam = Samantha part x 1
        2 = Sam = Samantha part x 2
        3 = Kate = Kate part x 1
        4 = Kate = Kate part x 2
        5 = Galen = Galen part x 1
        6 = Galen = Galen part x 2
        7 = Nina = Nina part x 1
        8 = Nina = Nina part x 2

R:
1. 2004787 = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FBB
        1 = Daryl = Daryl part x 1
        2 = Daryl = Daryl part x 2
        3 = Gustafa = Gustafa part x 1
        4 = Gustafa = Gustafa part x 2
        5 = Cody = Cody part x 1
        6 = Cody = Cody part x 2
        7 = Kassey = Kassey part x 1
        8 = Kassey = Kassey part x 2

R:
1. 2004788 = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FBC
        1 = Patrick = Patrick part x 1
        2 = Patrick = Patrick part x 2
        3 = Murray = Murray part x 1
        4 = Murray = Murray part x 2
        5 = Tim = Tim part x 1
        6 = Tim = Tim part x 2
        7 = Ruby = Ruby part x 1
        8 = Ruby = Ruby part x 2

R:
1. 2004789 = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FBD
        1 = Nancy = Nami part x 1
        2 = Nancy = Nami part x 2
        3 = Rock = Rock part x 1
        4 = Rock = Rock part x 2
        5 = Garcia = Griffin part x 1
        6 = Garcia = Griffin part x 2
        7 = Muffy = Muffy part x 1
        8 = Muffy = Muffy part x 2
```

```
R:
1. 200478A = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FBE
        1 = Carter = Carter part x 1
        2 = Carter = Carter part x 2
        3 = Flora = Flora part x 1
        4 = Flora = Flora part x 2
        5 = Vesta = Vesta part x 1
        6 = Vesta = Vesta part x 2
        7 = Marlin = Marlin part x 1
        8 = Marlin = Marlin part x 2

R:
1. 200478B = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FBF
        1 = Celia = Celia part x 1
        2 = Celia = Celia part x 2
        3 = Hardy = Hardy part x 1
        4 = Hardy = Hardy part x 2
        5 = Van = Van part x 1
        6 = Van = Van part x 2
        7 = Mooky = Mooky part x 1
        8 = Mooky = Mooky part x 2

R:
1. 200478C = Mary's Library.2nd Floor.Center Shelf Books
2. 2006FC0
        1 = David = Nak part x 1
        2 = David = Nak part x 2
        3 = Nic = Nic part x 1
        4 = Nic = Nic part x 2
        5 = Brad = Flak part x 1
        6 = Brad = Flak part x 2
        7 =
        8 =

1. 2004799 = Forget-Me-Not Valley References
2. 2006FCD
        1 =
        2 =
        3 =
        4 = Lou
        5 = Lu
        6 = Ellen
        7 = Barley
        8 = Rick

1. 200479A = Forget-Me-Not Valley References and recipes
2. 2006FCE
        1 = Thomas
        2 = Jeff
        3 = Carter
        4 = Gotz
        5 = Sprites
        6 =
        7 =
        8 = Recipes Lu has told you x 1

R:
```

```
1. 200479B = Recipes information
2. 2006FCF
        1 = Recipes Lu has told you x 2
        2 = Recipes Lu has told you x 4
        3 = Recipes Lu has told you x 8
        4 = Times Lu will tell you a recipe x 1
        5 = Times Lu will tell you a recipe x 2
        6 = Times Lu will tell you a recipe x 4
        7 = Times Lu will tell you a recipe x 8
        8 = Recipes Lu has left x 1

R:
1. 200479C = Recipes information and visitations
2. 2006FD0
        1 = Recipes Lu has left x 2
        2 = Recipes Lu has left x 4
        3 = Recipes Lu has left x 8
        4 = Lou visits
        5 = Lu visits
        6 =
        7 =
        8 =

R:
1. 200479E = Link-up star points
2. 2006FD2
        1 =
        2 = Link-up star points x 1
        3 = Link-up star points x 2
        4 = Link-up star points x 4
        5 = Link-up star points x 8
        6 = Link-up star points x 16
        7 = Link-up star points x 32
        8 = Link-up star points x 64

R:
1. 200479F = Link-up star points
2. 2006FD3
        1 = Link-up star points x 128
        2 =
        3 =
        4 =
        5 =
        6 =
        7 =
        8 =

1. 20047B4
2. 2006FE8
        1 =
        2 = Lu gave you a recipe this week
        3 =
        4 =
        5 =
        6 =
        7 =
        8 =

R:
1. 20047B8 - ????? = Transfer Name 1 (Farm)
```

```
2. 2006FEC - ?????

R:
1. 20047C8 - ????? = Transfer Name 2 (You)
2. 2006FFC - ?????

R:
1. 20047D8 - ????? = Transfer Name 3 (Child)
2. 200700C - ?????

--------------------------------------
12. POTENTIAL CODE BREAKER CODES
--------------------------------------

-------------------
A. Quick introduction
-------------------
```

This is probably the most useful section for anyone who is too lazy to create their own codes from all the above information. When I first made this FAQ, I was considering whether or not I should add this section, but I thought I should give this FAQ a "test drive" first. After all, I didn't want to make a complete FAQ that would not have been posted anyways.

If you do not understand section 4 on side effects, do not use these codes. Do not e-mail me about these codes either. If one of these is causing you a side effect, read sections 5 - 10 and make your own, or simply use a different value for the codes.

```
-------------------
B. Format of these codes
-------------------
```

All these codes are for Code Breaker, and they all are 8 bit codes. If you want to change them to 16 bit or to a different format, search for a different FAQ that would explain how this can be done.

The format is very simple (once you understand this at least):

```
3xxxxxxx 00nn
a b c d e f g h
yy
```

The '3' is the Code Breaker 8 bit code type.

The 'xxxxxxx' is the address. They are the 7 digit addresses found on sections 5 - 11.

The '00' on '00nn' is used for 8 bit code types.

The 'nn' in '00nn' is of course left as a variable so this can be more obvious. Simply replace 'nn' by 'yy' to get the complete code.

The 'yy' is the value calculated by assuming:
1. All the bits required to be set are set
2. All the bits required to be cleared are cleared
3. All unknown values are CLEARED. This is where side effects kick in.
4. For non-required but known bits that collide with the required ones, they are changed to whatever the game would normally require. These will not cause any side effects as they are because I know what they are for and what

they should ideally be.

The value yy would be optimal in the sense that only the Required ('R') bits were altered, and all other bits are assumed to be CLEARED. If these codes are used at the beginning of the game (after the game is started and you can begin to move around) then either ALL or MOST of the other bits should be cleared. The beginning is therefore the ideal time to use these codes.

Remember to remove the codes immediately after putting them in so that all other bits (and the relevant bits as well) can change normally as the game requires.

The 'a b c d e f g h' are each of the 8 bits in the address.
1. To make conversions easier, the 8 bits are in big-endian order, meaning 'a' is the most significant bit and 'h' is the least significant bit. Yes, this is the reverse order I explained in the other sections. I did this in big-endian on purpose to make conversions from binary to hexadecimal easier.
2. A '0' in a bit means it should be cleared on the code.
3. A '1' in a bit means it should be set on the code.
4. A '?' in a bit means it is an unknown. Clear it or set it at your own risk! In all the values I provided, they are all cleared.

If the values I provided are giving you side effects, simply replace some of the '?' with a '1' and based on that stream calculate a new value to use. For example, if I have

```
1 0 ? 0 ? 1 1 0
```

and for the '?' you select to set the first '?' (1) and clear the second '?', (0) then your new bit stream is:

```
1 0 1 0 0 1 1 0 = 10100110 = A6 in hexadecimal.
```

It's that simple. Remember, you only need to do this step if you don't want to use the value I have provided. In the value I have provided, all the '?' are cleared (i.e. they are '0') All bits without a '?' should be exactly what it says they are.

Finally, in case anyone can't move the 2 digit 'yy' value into 'xx', at the end of each section I will write out the complete codes using the value I provided. And to reiterate, these are the only codes Required ('R') to receive all the bonuses.

Stage 1 codes are written as "1. 3xxxxxxx 00yy" and stage 2 codes are written as "2. 3xxxxxxx 00yy". Remember, ONLY ONE of the 2 stages is needed, NOT both! For the transferred names, however, I just put each stage code segment on its own pile.

```
-------------------
C. From 5. Link Up Stars and the Seaside Cottage
-------------------
```

Link-up star points:

```
1. 3200479E 00xx
2. 32006FD2 00xx
0 0 0 1 0 1 0 ?
14

1. 3200479F 00xx
```

```
2. 32006FD3 00xx
? ? ? ? ? ? ? 1
01

Absolute Codes:

Stage 1:
3200479E 0014
3200479F 0001

Stage 2:
32006FD2 0014
32006FD3 0001

-------------------
D. From 6. Lou: Artifacts Collector,
        7. Lu: The Recipes Girl and
        8. Forget-Me-Not Valley Villagers' References
-------------------

This part covers codes from sections 6 - 8 because the addresses used for
each of those sections is also used for data in others. Section 8 is not
required, but since it collides with data from section 7, I had to add at
least part of it. So instead I added all of it, and the values I used
provide the player with all those bonuses as well.

Forget-Me-Not Valley References:

1. 32004799 00xx
2. 32006FCD 00xx
1 1 1 1 1 ? ? ?
F8

Forget-Me-Not Valley References and recipes:

1. 3200479A 00xx
2. 32006FCE 00xx
0 ? ? 1 1 1 1 1
1F

Recipes information:

1. 3200479B 00xx
2. 32006FCF 00xx
0 1 0 1 0 0 0 0
50

Recipes information and visitations:

1. 3200479C 00xx
2. 32006FD0 00xx
? ? ? 1 1 1 0 1
1D

Records sold:

1. 320045D2 00xx
2. 32006E06 00xx
0 0 0 0 1 0 1 0
0A
```

```
Absolute Codes:

Stage 1:
32004799 00F8
3200479A 001F
3200479B 0050
3200479C 001D
320045D2 000A

Stage 2:
32006FCD 00F8
32006FCE 001F
32006FCF 0050
32006FD0 001D
32006E06 000A

-------------------
E. From 9. Transferred Names
-------------------

The actual transferred names' maximum letters is at most 15, since the name
must be null terminated. However, since I don't know the actual max that is
naturally allowed, I have provided all 16 addresses. All letters for these
names are in standard ASCII format. Find an ASCII table somewhere if you
don't know what letter is represented by each value.

Transfer Name 1 (Farm) Letters 1 - 16

Stage 1:
320047B8 00xx
320047B9 00xx
320047BA 00xx
320047BB 00xx
320047BC 00xx
320047BD 00xx
320047BE 00xx
320047BF 00xx
320047C0 00xx
320047C1 00xx
320047C2 00xx
320047C3 00xx
320047C4 00xx
320047C5 00xx
320047C6 00xx
320047C7 00xx

Stage 2:
32006FEC 00xx
32006FED 00xx
32006FEE 00xx
32006FEF 00xx
32006FF0 00xx
32006FF1 00xx
32006FF2 00xx
32006FF3 00xx
32006FF4 00xx
32006FF5 00xx
32006FF6 00xx
32006FF7 00xx
```

```
32006FF8 00xx
32006FF9 00xx
32006FFA 00xx
32006FFB 00xx

Transfer Name 2 (You) Letters 1 - 16

Stage 1:
320047C8 00xx
320047C9 00xx
320047CA 00xx
320047CB 00xx
320047CC 00xx
320047CD 00xx
320047CE 00xx
320047CF 00xx
320047D0 00xx
320047D1 00xx
320047D2 00xx
320047D3 00xx
320047D4 00xx
320047D5 00xx
320047D6 00xx
320047D7 00xx

Stage 2:
32006FFC 00xx
32006FFD 00xx
32006FFE 00xx
32006FFF 00xx
32007000 00xx
32007001 00xx
32007002 00xx
32007003 00xx
32007004 00xx
32007005 00xx
32007006 00xx
32007007 00xx
32007008 00xx
32007009 00xx
3200700A 00xx
3200700B 00xx

Transfer Name 3 (Child) Letters 1 - 16

Stage 1:
320047D8 00xx
320047D9 00xx
320047DA 00xx
320047DB 00xx
320047DC 00xx
320047DD 00xx
320047DE 00xx
320047DF 00xx
320047E0 00xx
320047E1 00xx
320047E2 00xx
320047E3 00xx
320047E4 00xx
320047E5 00xx
```

```
320047E6 00xx
320047E7 00xx

Stage 2:
3200700C 00xx
3200700D 00xx
3200700E 00xx
3200700F 00xx
32007010 00xx
32007011 00xx
32007012 00xx
32007013 00xx
32007014 00xx
32007015 00xx
32007016 00xx
32007017 00xx
32007018 00xx
32007019 00xx
3200701A 00xx
3200701B 00xx

Absolute Codes:
None. I don't know what letters you want for your transferred names.

-------------------
F. From 10. The Library: Second Floor’s Mysterious Book Shelf
-------------------

For the first book entry, bits 7 and 8 are unknown, but I strongly suspect
they correspond to a 37th book, which contains information about your child
from AWL. Therefore, I have chosen to set those bits (the bits set on
parenthesis: '(1)') in case I ever find that book.

Because some entries do not have 3-part information in them, all bits don't
need to be set. This, however, seems to cause no side effects whatsoever.
In any case, this is the only section where I have set more bits than
needed. Feel free to try different values for a more 'optimal' result.

Since I forgot what the max value for each entry is, and since I don't want
to look through all the books again, I did not provide the bit sequence used
for the intermediate entries (i.e. all other entries except the first and
last) since I set all the bits without needing to set them all.

Book Entries:

1. 32004783 00xx
2. 32006FB7 00xx
(1) (1) 1 1 ? ? ? ?
F0

1. 32004784 00xx
2. 32006FB8 00xx
FF

1. 32004785 00xx
2. 32006FB9 00xx
FF

1. 32004786 00xx
2. 32006FBA 00xx
```

```
FF

1. 32004787 00xx
2. 32006FBB 00xx
FF

1. 32004788 00xx
2. 32006FBC 00xx
FF

1. 32004789 00xx
2. 32006FBD 00xx
FF

1. 3200478A 00xx
2. 32006FBE 00xx
FF

1. 3200478B 00xx
2. 32006FBF 00xx
FF

1. 3200478C 00xx
2. 32006FC0 00xx
? ? 1 1 1 1 1 1
3F

Absolute Codes:

Stage 1:
32004783 00F0
32004784 00FF
32004785 00FF
32004786 00FF
32004787 00FF
32004788 00FF
32004789 00FF
3200478A 00FF
3200478B 00FF
3200478C 003F

Stage 2:
32006FB7 00F0
32006FB8 00FF
32006FB9 00FF
32006FBA 00FF
32006FBB 00FF
32006FBC 00FF
32006FBD 00FF
32006FBE 00FF
32006FBF 00FF
32006FC0 003F

--------------------------------------
13. Q & A
--------------------------------------

Q: Why won’t my code work?

A: Read section 4 part C. It explains about the different versions. There
```

is also a chance you may have decided on a value at the address that’s not correct. It is also possible that there might be another format still, (i.e. another Stage) but I doubt that was the problem. All these addresses correspond to the U.S. version as well.

Q: I used a code and now I’m getting side effects! Your addresses must not be working correctly!

A: Read section 4 parts A and B. It describes why side effects are difficult to work around with. Also, read the introduction which has additional information about this.
Furthermore, there are such things as non-code side effects resulting from a code. For instance, give yourself 127 1-ups in Mario 1 for the NES and after you get an additional 1-up you will die. Normally, you might think this is a side effect caused by the code itself, but it's not. It is a program side effect. 127 lives + 1 more 1-up = 128, and 128 as a byte is 10000000 in binary, which is also negative 128! A 1 in the most significant digit of a byte, when interpreted as a signed number, indicates it is negative. So when you get another 1-up the game thinks you got negative life and you die!
This kind of side effect cannot be blamed on a code, however, you just have to realize this kind of situation and figure out what to do about it. I don't know of any program side effects for this game at this point, but it is possible, nonetheless. Remember the programmers wrote the game under the assumption that you will not cheat, so they are not going to worry too much about side effects in the program resulting from codes.

--------------------------------------
## 14. CREDITS
--------------------------------------

As I mentioned already, all of these link-up addresses were independently discovered. However, I received information about the game from different people so that I could know what to look for. In compiling the credits list, I went through the code topic on gamefaqs and gave credit to those that made such contributions, according to what was posted in that topic.

CJayC - Of course as always, special thanks to him for GameFaqs and for hosting my FAQ.

Stephen Ng - For hosting my FAQ on ign.com.

Leo Chan - For hosting my FAQ on neoseeker.

Cherubae - Has the best site about this game I’ve seen. His site informed me about many of the bonuses you are supposed to get in this game.

vjay8088 and Chaoswolf82 - First persons to discover that memory addresses change when a save game is reloaded. vjay8088 came up with the term “Stage” to refer to each different possible set of data addressing. The different Stages mean that even addresses are not completely universal in this game. He was also the first to post an all houses code that gives you the seaside cottage as well, and to post a code that would also give you the record player (since the record player is in the same byte as other house improvement data).

hailstorm - First user to request link-up codes in the code topic. Without this request, I would have never even searched for these!

irrimn - Another user who requested link-up codes, whose request I saw.

```
He also informed me of many details like how to check your current link-up
stars value, where you’re supposed to buy the records, etc.

exdeath18o - For the link to Cherubae’s site.

mvegeta - For the save file that eventually got me to find Lu’s recipes
address.

rou dai buushin and eldiablos - Although a lot of users requested this
before them, their request for an infinite water in watering can, which
got me to hack it myself, eventually led me to discover that data in this
game is not byte aligned. Since that was a very important issue, they
deserve some mentioning.

JamBun - Gave me the idea to make a link-up codes FAQ. He also told me that
the characters’ information from AWL that are suppose to appear in this
game might have something to do with the library’s second floor bookshelf,
which led to that discovery.

ShAyA - For harrassing me to do the stage 2 codes. Hopefully this will help
out alot of others as well.
```