CS3723 Pgm Assignment #1: Storage Management in C (60 pts)

In this assignment, you will create **storage management functions** which manage memory and implement garbage collection. Your code must understand metadata to support the overall garbage collection process. Why is that necessary?

There are several important phases of memory management:

**Initialize Phase** Initialize the memory to one huge free node. The storage manager's pFreeHead will then point to the starting address in that huge node having a cGC of 'F' (free). Although the free memory list initially has only one node, it can have many free nodes after a Garbage Collection Phase completes. The initialize phase only happens once - prior to any usage.

**Usage Phase** In this phase:

* When memory allocation is requested, your memory allocation function will return a suitable "In Use" node from a free node.
* The "In Use" node will be marked with a cGC of 'U' (in use).
* Your memory allocation function might return the left over portion of the free node to the free list or include it in the "In Use" node. (see below)
* In this phase, we will also associate nodes to each other.
* Other than returning the left over portion of an allocation request to the free list, we will NOT free any nodes (i.e., we won't change any nodes from 'U' to 'F') during the Usage Phase.

**GC Phase** There are three subphases to the Garbage Collection Phase :

**Mark Subphase** Walk through all memory from the beginning and mark **all** nodes to a cGC of 'C' (candidate). It is easier to combine adjacent free areas if all nodes are marked as 'C'.

**Follow Subphase** From any valid starting point (which we are simulating with a hash table), follow the nodes based on metadata and mark each referenceable node as 'U' (in use).

**Collection Subphase** We will build the free list from the entire memory. (We are affectively ignoring the old free linked list. Discard any knowledge of it.) Walk through all the memory looking for 'C' nodes, combining adjacent ones into a single free area. Each free area will be placed on the front of the new free list.

To help reduce the code that you have to write (allowing you to focus on the most important concepts) and to reduce debugging difficulty, I have provided the following:

cs3723p1Driver.c - reads the input file, calls your storage management functions, and uses a hash table to store the addresses of allocated memory (so that those can be subsequently associated/disassociated with other nodes). An important function is the setData function which uses metadata to set attributes in a node. The driver also provides functions for printing the heap memory to help with debugging.

cs3723p1.h - include file for this program. Some important typedefs:

NodeType – describes one node type: name, beginning subscript in metaAttrM array, total size

MetaAttr – describes one attribute in a node type: name, type, size in bytes, and offset

InUseNode - contains the node's size, node type, cGc, and the user's data

StorageManager - a structure that contains the address of the heap (pBeginStorage), pointer to the end of the heap (to the address that immediately follows it), pointer to the head of the free memory list, an array of NodeType entries, and an array of MetaAttr entries. It does not have a count of the number of entries in those arrays. Instead, sentinels are used to mark the end of the arrays.

MMResult - used by the mm... functions to specify whether they executed successfully.

FreeNode - the first portion of a free node.

hashApi.cpp - C++ code to integrate C with the C++ Hash Table Class (unordered\_map). This supports functions getHash, putHash, eraseAll, and getAll. This is only used by the driver.

hexDump64.o - A function to produce hexDumps of data. This is used by the driver when printing the heap's contents. This file is available on the server at this location: **/usr/local/courses/clark/cs3723/2017Fa**. Copy that .o to your folder.

printNode.o - A function to print the contents of a node, showing each attribute and its values. This is used by the driver when printing the heap's contents. This file is available on the server at this location: **/usr/local/courses/clark/cs3723/2017Fa**. Copy that .o to your folder.

p1Input.txt - Input text file suitable for the driver. The driver uses **stdin** so redirect input from this file.

makefile - Please use this makefile to create your **p1** executable. Note that you should not use the hen servers. The makefile uses **g++** instead of gcc. This file can be found at **/usr/local/courses/clark/cs3723/2017Fa**. To build the executable (with it automatically building the other pieces), type  
 **make p1**

**You will need to code** the following functions; however, due to modularity concerns, you may need to create 1 or more extra functions. Your code should be placed in cs3723p1.c.

void **mmInit**(StorageManager \*pMgr, MMResult \*pmmResult)

This is part of the Init Phase. The **driver** already set pMgr->pBeginStorage to the address of the heap, pMgr->iHeapSize, pMgr->pEndStorage, and pMgr->iMinimumNodeSize. Initialize the memory to one huge free node. You need to set pMgr->pFreeHead to point to the starting address of that huge node.

* Zero out the entire heap.
* Set the cGC to 'F'
* Set the shNodeSize

void \* **mmAllocate**(StorageManager \*pMgr, short shDataSize, short shNodeType, char sbData[]

, MMResult \*pmmResult)

This is part of the Usage Phase. This is satisfying a memory allocation request using the first node from the free list that is big enough.

* It is passed the user data size (which is less than the size to allocate due to the node overhead of size, node type, and cGC), the node type, and the user's data. How much do we allocate? the user data size plus ?
* Searches for a free node with enough memory to satisfy the needed size. If there isn't a large enough free node to satisfy the request:
  + Set pmmResult->rc to RC\_NOT\_AVAIL.
  + Copy an error message to pmmResult->szErrorMessage.
  + Functionally return NULL.
* The remaining bullets assume there was a large enough free node to satisfy the request.
* Remove that free node from the free list, updating the free list. Do this regardless of whether there is enough free space to make a free node
* The new InUseNode will **begin** at the address of the free node.
* Determine whether there is enough remaining space in the free node to make that remaining space a new free node (see pMgr->iMininumNodeSize). If there isn't enough space left, change the allocated size to be the free node's size. Otherwise, add the new free node (which uses the remaining space) to the free list. (We will **ALWAYS** add to the **FRONT** of the free list.)
* Initializes an InUseNode:
  + Set its cGc to 'U'.
  + Set its node type.
  + Set its size.
  + Set its sbData.
* Returns a pointer (from the user's perspective) to the allocated memory. This is not the address of the InUseNode

void **mmMark**(StorageManager \*pMgr, MMResult \*pmmResult)

This is the first subphase of Garbage Collection. Beginning with the start of the heap, sequentially mark every node as 'C'. Look at the driver's smDump function to see how to advance through the heap touching every adjacent node.

void **mmFollow**(StorageManager \*pMgr, void \*pUserData, MMResult \*pmmResult)

The Follow subphase of Garbage Collection is handled by the driver and **mmFollow**. The driver will actually call this function multiple times during this subphase. It is passed a user data pointer.

* If the current node's cGC is 'U', it has already been followed; therefore, return.
* Otherwise, set its cGC to 'U'. Recursively follow each non-null pointer referenced by this node. Your code must make certain that a cycle in the data doesn't cause an infinite loop. This code has to understand metadata to know where the pointers are located.

void **mmCollect**(StorageManager \*pMgr, MMResult \*pmmResult)

This is the third subphase of Garbage Collection. In this phase, you will sequentially traverse the heap, collecting the 'C' nodes, combining adjacent 'C' nodes, and placing the nodes onto a new free list. Each insertion will be to the **front** of the free list. As you collect free space, print one of the two following messages:

printf("\tCollecting %08lX\n", ULAddr(pCandidate));

printf("\tCombining %08lX with %08lX\n"

, ULAddr(pPrecedes), ULAddr(pCandidate));

void **mmAssoc**(StorageManager \*pMgr, void \*pUserDataFrom, char szAttrName[]

, void \*pUserDataTo, MMResult \*pmmResult)

sets a user pointer in the specified user data node to a new referenced user data node.

* Search for the specified attribute name in the meta data for the **from** node.
* If not found, set the pmmResult->rc to RC\_ASSOC\_ATTR\_NOT\_FOUND, provide an error message that contains the attribute name, and return. (Ignore the remaining bullets.)
* If the specified attribute is not apointer, set the pmmResult->rc to RC\_ASSOC\_ATTR\_NOT\_PTR provide an error message that contains the attribute name, and return. (Ignore the remaining bullets.)
* Change the user pointer in the specified user data node to point to the new referenced user data node or NULL (if that was specified).
* Hint: once you know the offset in the user data:

void \*\*ppNode = (void \*\*)&(pInUseFrom->sbData[pAttr->shOffset]);

\*ppNode = pUserDataTo;

Notes:

1. To help understand how to use the metadata, examine the driver's setData function.
2. To simplify grading, please include your C code and output (generated on a **fox** server) in a zip file with your name in mixed case with last name followed by first (e.g., KingJoe.zip).
3. Your code must follow my **programming standards**.
4. You must make certain your code works on a fox server and can be compiled by the specified makefile.
5. For **Microsoft Visual Studio** Users:

* Since I provided the hexDump64 and printNode as .o files, you can't use them. Inside the driver, rename dumbHexDump to hexDump and dumbPrintNode to printNode. Please remember to delete those before running on the UTSA Linux server.
* If you need a wider Console Window:
  + Once the console window displays (you may want a break point in your code so that it doesn't disappear), click the top left corner of the console window.
  + Properties
  + Layout
  + Change the Screen Buffer Size to 120
  + Change the Window Size to 120

**Sample Partial Output:**

Metadata

Node Type Beg Attr Sub Total Sz

Customer 0 48

Attribute Name Type Offset Size

customerId S 0 8

name S 8 16

pFirstItem P 24 8

pNextCust P 32 8

balance D 40 8

LineItem 5 32

Attribute Name Type Offset Size

productId S 0 10

iQtyReq I 12 4

dCost D 16 8

pNextItem P 24 8

>>> ALLOC C111 Customer 111,Sal A Mander,NULL,NULL,100.00

>>> PRTNODE C111

Alloc Address Size Node Type Data Address

0x2601070 53 0 02601075

Attr Name Type Value

customerId S 111

name S Sal A Mander

pFirstItem P 00000000

pNextCust P 00000000

balance D 100.000000

>>> ALLOC C222 Customer 222,Barb Wire,NULL,NULL,200.00

>>> ALLOC PPF001 LineItem PPF001,5,9.95,NULL

>>> PRTALL

Alloc Address Size Node Type Data Address

0x26010da 37 1 026010DF

Attr Name Type Value

productId S PPF001

iQtyReq I 5

dCost D 9.950000

pNextItem P 00000000

Alloc Address Size Node Type Data Address

0x26010a5 53 0 026010AA

Attr Name Type Value

customerId S 222

name S Barb Wire

pFirstItem P 00000000

pNextCust P 00000000

balance D 200.000000

Alloc Address Size Node Type Data Address

0x2601070 53 0 02601075

Attr Name Type Value

customerId S 111

name S Sal A Mander

pFirstItem P 00000000

pNextCust P 00000000

balance D 100.000000

\*

\* associate customer 111 with a next pointing to 222

\*

>>> ASSOC C111 pNextCust C222

\* customer 111's pNextCust should point to 222

>>> PRTNODE C111

Alloc Address Size Node Type Data Address

0x2601070 53 0 02601075

Attr Name Type Value

customerId S 111

name S Sal A Mander

pFirstItem P 00000000

pNextCust P 026010AA

balance D 100.000000

\*

\* associate customer 111 to PPF001

\*

>>> ASSOC C111 pFirstItem PPF001

>>> PRTNODE C111

Alloc Address Size Node Type Data Address

0x2601070 53 0 02601075

Attr Name Type Value

customerId S 111

name S Sal A Mander

pFirstItem P 026010DF

pNextCust P 026010AA

balance D 100.000000

...

\*

\* Add many customers and line items

\*

>>> ALLOC C888 Customer 888,Joe King,NULL,NULL,88.00

>>> ALLOC C911 Customer 911,Lee King,NULL,NULL,91.00

>>> ALLOC C922 Customer 922,Mae King,NULL,NULL,92.00

>>> ALLOC MCW001 LineItem MCW001,1,12.45,NULL

>>> ALLOC C933 Customer 933,Ray King,NULL,NULL,93.00

>>> ALLOC C944 Customer 944,Freida King,NULL,NULL,94.00

>>> ALLOC SSX001 LineItem SSX001,1,29.95,NULL

>>> ASSOC MCW001 pNextItem SSX001

>>> ALLOC TTP001 LineItem TTP001,2,9.95,NULL

>>> DEREF SSX001

>>> DUMP

Address Mem Size NodeType

02601070 InUse 53 0

02601075 1 1 1 . . . . . S a l A M a n d e r

31 31 31 00 00 00 00 00 53 61 6C 20 41 20 4D 61 6E 64 65 72

02601089 . . . . . . ` . . . . . . . ` . . . . .

00 00 00 00 DF 10 60 02 00 00 00 00 AA 10 60 02 00 00 00 00

0260109D . . . . . . Y @

00 00 00 00 00 00 59 40

026010A5 InUse 53 0

026010AA 2 2 2 . . . . . B a r b W i r e . . .

32 32 32 00 00 00 00 00 42 61 72 62 20 57 69 72 65 00 00 00

026010BE . . . . . . . . . . . . . . ` . . . . .

00 00 00 00 00 00 00 00 00 00 00 00 04 11 60 02 00 00 00 00

026010D2 . . . . . . i @

00 00 00 00 00 00 69 40

026010DA InUse 37 1

026010DF P P F 0 0 1 . . . . . . . . . . f f f f

50 50 46 30 30 31 00 00 00 00 00 00 05 00 00 00 66 66 66 66

026010F3 f . # @ . . . . . . . .

66 E6 23 40 00 00 00 00 00 00 00 00

026010FF InUse 53 0

02601104 3 3 3 . . . . . M i s t y W i n d . .

33 33 33 00 00 00 00 00 4D 69 73 74 79 20 57 69 6E 64 00 00

02601118 . . . . . . . . . . . . 9 . ` . . . . .

00 00 00 00 00 00 00 00 00 00 00 00 39 11 60 02 00 00 00 00

0260112C . . . . . . Q @

00 00 00 00 00 80 51 40

02601134 InUse 53 0

02601139 4 4 4 . . . . . E m o r y B o a r d .

34 34 34 00 00 00 00 00 45 6D 6F 72 79 20 42 6F 61 72 64 00

0260114D . . . . . . . . . . . . " . ` . . . . .

00 00 00 00 00 00 00 00 00 00 00 00 22 12 60 02 00 00 00 00

02601161 . . . . Q 8 F @

B8 1E 85 EB 51 38 46 40

02601169 InUse 53 0

0260116E 5 5 5 . . . . . H o l l y G r a p h .

35 35 35 00 00 00 00 00 48 6F 6C 6C 79 20 47 72 61 70 68 00

02601182 . . . . . . ` . . . . . . . ` . . . . .

00 00 00 00 A3 11 60 02 00 00 00 00 AA 10 60 02 00 00 00 00

02601196 . . . . . . K @

00 00 00 00 00 80 4B 40

0260119E InUse 37 1

026011A3 S B B 0 0 1 . . . . . . . . . . 3 3 3 3

53 42 42 30 30 31 00 00 00 00 00 00 02 00 00 00 33 33 33 33

026011B7 3 . 3 @ . . ` . . . . .

33 F3 33 40 C8 11 60 02 00 00 00 00

026011C3 InUse 37 1

026011C8 B O M 0 0 1 . . . . . . . . . . 3 3 3 3

42 4F 4D 30 30 31 00 00 00 00 00 00 01 00 00 00 33 33 33 33

026011DC 3 . = @ . . . . . . . .

33 F3 3D 40 00 00 00 00 00 00 00 00

026011E8 InUse 53 0

026011ED 6 6 6 . . . . . B o b L H e a d . .

36 36 36 00 00 00 00 00 42 6F 62 20 4C 20 48 65 61 64 00 00

02601201 thru 02601214 (1 lines) suppressed - all zero

02601215 . . . . . . P @

00 00 00 00 00 80 50 40

0260121D InUse 53 0

02601222 7 7 7 . . . . . F l o N W a t e r .

37 37 37 00 00 00 00 00 46 6C 6F 20 4E 20 57 61 74 65 72 00

02601236 thru 02601249 (1 lines) suppressed - all zero

0260124A . . . . . @ S @

00 00 00 00 00 40 53 40

02601252 InUse 53 0

02601257 8 8 8 . . . . . J o e K i n g . . . .

38 38 38 00 00 00 00 00 4A 6F 65 20 4B 69 6E 67 00 00 00 00

0260126B thru 0260127E (1 lines) suppressed - all zero

0260127F . . . . . . V @

00 00 00 00 00 00 56 40

02601287 InUse 53 0

0260128C 9 1 1 . . . . . L e e K i n g . . . .

39 31 31 00 00 00 00 00 4C 65 65 20 4B 69 6E 67 00 00 00 00

026012A0 thru 026012B3 (1 lines) suppressed - all zero

026012B4 . . . . . . V @

00 00 00 00 00 C0 56 40

026012BC InUse 53 0

026012C1 9 2 2 . . . . . M a e K i n g . . . .

39 32 32 00 00 00 00 00 4D 61 65 20 4B 69 6E 67 00 00 00 00

026012D5 thru 026012E8 (1 lines) suppressed - all zero

026012E9 . . . . . . W @

00 00 00 00 00 00 57 40

026012F1 InUse 37 1

026012F6 M C W 0 0 1 . . . . . . . . . . f f f f

4D 43 57 30 30 31 00 00 00 00 00 00 01 00 00 00 66 66 66 66

0260130A f . ( @ . . ` . . . . .

66 E6 28 40 85 13 60 02 00 00 00 00

02601316 InUse 53 0

0260131B 9 3 3 . . . . . R a y K i n g . . . .

39 33 33 00 00 00 00 00 52 61 79 20 4B 69 6E 67 00 00 00 00

0260132F thru 02601342 (1 lines) suppressed - all zero

02601343 . . . . . @ W @

00 00 00 00 00 40 57 40

0260134B InUse 53 0

02601350 9 4 4 . . . . . F r e i d a K i n g .

39 34 34 00 00 00 00 00 46 72 65 69 64 61 20 4B 69 6E 67 00

02601364 thru 02601377 (1 lines) suppressed - all zero

02601378 . . . . . . W @

00 00 00 00 00 80 57 40

02601380 InUse 37 1

02601385 S S X 0 0 1 . . . . . . . . . . 3 3 3 3

53 53 58 30 30 31 00 00 00 00 00 00 01 00 00 00 33 33 33 33

02601399 3 . = @ . . . . . . . .

33 F3 3D 40 00 00 00 00 00 00 00 00

026013A5 InUse 37 1

026013AA T T P 0 0 1 . . . . . . . . . . f f f f

54 54 50 30 30 31 00 00 00 00 00 00 02 00 00 00 66 66 66 66

026013BE f . # @ . . . . . . . .

66 E6 23 40 00 00 00 00 00 00 00 00

026013CA Free 42

Next:00000000

...

\*

\* This line item is smaller than the last free node;

\* however, the remaining free size is too small for a

\* free node. We give NHC001 all of the free node's

\* size as its size.

>>> ALLOC NHC001 LineItem NHC001,1,9.95,NULL

...

Alloc Address Size Node Type Data Address

0x26013ca 42 1 026013CF

Attr Name Type Value

productId S NHC001

iQtyReq I 1

dCost D 9.950000

pNextItem P 00000000

...

\*

\* Remove our reference to the beginning of the linked list

\*

>>> DEREF MCW001

\*

\* Do a Garbage Collection which will gain two free areas

\*

>>> GCOLL

Collecting 026012F1

Collecting 02601380

>>> DUMP

Address Mem Size NodeType

02601070 InUse 53 0

02601075 1 1 1 . . . . . S a l A M a n d e r

31 31 31 00 00 00 00 00 53 61 6C 20 41 20 4D 61 6E 64 65 72

02601089 . . . . . . ` . . . . . . . ` . . . . .

00 00 00 00 DF 10 60 02 00 00 00 00 AA 10 60 02 00 00 00 00

0260109D . . . . . . Y @

00 00 00 00 00 00 59 40

026010A5 InUse 53 0

026010AA 2 2 2 . . . . . B a r b W i r e . . .

32 32 32 00 00 00 00 00 42 61 72 62 20 57 69 72 65 00 00 00

026010BE . . . . . . . . . . . . . . ` . . . . .

00 00 00 00 00 00 00 00 00 00 00 00 04 11 60 02 00 00 00 00

026010D2 . . . . . . i @

00 00 00 00 00 00 69 40

026010DA InUse 37 1

026010DF P P F 0 0 1 . . . . . . . . . . f f f f

50 50 46 30 30 31 00 00 00 00 00 00 05 00 00 00 66 66 66 66

026010F3 f . # @ . . . . . . . .

66 E6 23 40 00 00 00 00 00 00 00 00

026010FF InUse 53 0

02601104 3 3 3 . . . . . M i s t y W i n d . .

33 33 33 00 00 00 00 00 4D 69 73 74 79 20 57 69 6E 64 00 00

02601118 . . . . . . . . . . . . 9 . ` . . . . .

00 00 00 00 00 00 00 00 00 00 00 00 39 11 60 02 00 00 00 00

0260112C . . . . . . Q @

00 00 00 00 00 80 51 40

02601134 InUse 53 0

02601139 4 4 4 . . . . . E m o r y B o a r d .

34 34 34 00 00 00 00 00 45 6D 6F 72 79 20 42 6F 61 72 64 00

0260114D . . . . . . . . . . . . " . ` . . . . .

00 00 00 00 00 00 00 00 00 00 00 00 22 12 60 02 00 00 00 00

02601161 . . . . Q 8 F @

B8 1E 85 EB 51 38 46 40

02601169 InUse 53 0

0260116E 5 5 5 . . . . . H o l l y G r a p h .

35 35 35 00 00 00 00 00 48 6F 6C 6C 79 20 47 72 61 70 68 00

02601182 . . . . . . ` . . . . . . . ` . . . . .

00 00 00 00 A3 11 60 02 00 00 00 00 AA 10 60 02 00 00 00 00

02601196 . . . . . . K @

00 00 00 00 00 80 4B 40

0260119E InUse 37 1

026011A3 S B B 0 0 1 . . . . . . . . . . 3 3 3 3

53 42 42 30 30 31 00 00 00 00 00 00 02 00 00 00 33 33 33 33

026011B7 3 . 3 @ . . ` . . . . .

33 F3 33 40 C8 11 60 02 00 00 00 00

026011C3 InUse 37 1

026011C8 B O M 0 0 1 . . . . . . . . . . 3 3 3 3

42 4F 4D 30 30 31 00 00 00 00 00 00 01 00 00 00 33 33 33 33

026011DC 3 . = @ . . . . . . . .

33 F3 3D 40 00 00 00 00 00 00 00 00

026011E8 InUse 53 0

026011ED 6 6 6 . . . . . B o b L H e a d . .

36 36 36 00 00 00 00 00 42 6F 62 20 4C 20 48 65 61 64 00 00

02601201 thru 02601214 (1 lines) suppressed - all zero

02601215 . . . . . . P @

00 00 00 00 00 80 50 40

0260121D InUse 53 0

02601222 7 7 7 . . . . . F l o N W a t e r .

37 37 37 00 00 00 00 00 46 6C 6F 20 4E 20 57 61 74 65 72 00

02601236 thru 02601249 (1 lines) suppressed - all zero

0260124A . . . . . @ S @

00 00 00 00 00 40 53 40

02601252 InUse 53 0

02601257 8 8 8 . . . . . J o e K i n g . . . .

38 38 38 00 00 00 00 00 4A 6F 65 20 4B 69 6E 67 00 00 00 00

0260126B thru 0260127E (1 lines) suppressed - all zero

0260127F . . . . . . V @

00 00 00 00 00 00 56 40

02601287 InUse 53 0

0260128C 9 1 1 . . . . . L e e K i n g . . . .

39 31 31 00 00 00 00 00 4C 65 65 20 4B 69 6E 67 00 00 00 00

026012A0 thru 026012B3 (1 lines) suppressed - all zero

026012B4 . . . . . . V @

00 00 00 00 00 C0 56 40

026012BC InUse 53 0

026012C1 9 2 2 . . . . . M a e K i n g . . . .

39 32 32 00 00 00 00 00 4D 61 65 20 4B 69 6E 67 00 00 00 00

026012D5 thru 026012E8 (1 lines) suppressed - all zero

026012E9 . . . . . . W @

00 00 00 00 00 00 57 40

026012F1 Free 37

Next:00000000

02601316 InUse 53 0

0260131B 9 3 3 . . . . . R a y K i n g . . . .

39 33 33 00 00 00 00 00 52 61 79 20 4B 69 6E 67 00 00 00 00

0260132F thru 02601342 (1 lines) suppressed - all zero

02601343 . . . . . @ W @

00 00 00 00 00 40 57 40

0260134B InUse 53 0

02601350 9 4 4 . . . . . F r e i d a K i n g .

39 34 34 00 00 00 00 00 46 72 65 69 64 61 20 4B 69 6E 67 00

02601364 thru 02601377 (1 lines) suppressed - all zero

02601378 . . . . . . W @

00 00 00 00 00 80 57 40

02601380 Free 37

Next:026012F1

026013A5 InUse 37 1

026013AA T T P 0 0 1 . . . . . . . . . . f f f f

54 54 50 30 30 31 00 00 00 00 00 00 02 00 00 00 66 66 66 66

026013BE f . # @ . . . . . . . .

66 E6 23 40 00 00 00 00 00 00 00 00

026013CA InUse 42 1

026013CF N H C 0 0 1 . . . . . . . . . . f f f f

4E 48 43 30 30 31 00 00 00 00 00 00 01 00 00 00 66 66 66 66

026013E3 f . # @ . . . . . . . . . . . . .

66 E6 23 40 00 00 00 00 00 00 00 00 00 00 00 00 00