CS3723 Pgm Assignment #1: Storage Management in C (50 pts)

In this assignment, you will create **heap storage management functions** for initializing the heap, allocating nodes, and freeing nodes. As discussed during the Storage Management lectures, your software must be capable of managing different sized allocations. Since free items and allocated items are in the same memory, you must keep track of the free nodes in a doubly linked list of free nodes (using the memory in the free node). This doubly linked list MUST NOT be in separate memory from the heap being managed (if it is you will lose at least 80% of the points).

To help reduce the code that you have to write 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 freed). The driver also provides functions for printing the heap memory to help with debugging.

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

FreeNode - contains its size (at the beginning), free flag, previous free node pointer, next free node pointer, a variable number of unused bytes, and its size (at the end).

AllocNode - contains its size (at the beginning), allocated flag, the data for the allocated node, and it size (at the end)

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, minimum node size.

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/cs3723/2018Sp/P1**. Copy that .o to your folder.

p1Input.txt - Input text file suitable for the driver. The driver uses **stdin** so redirect 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/cs3723/2018Sp/P1**. To build the executable (with it automatically building the other pieces), type  
 **make p1**

You can see more information about the make utility on my website.

You will need to **code** the following C functions; however, due to modularity concerns, you will probably create extra functions. Look at the typedefs in the include file.

void **smInit**(StorageManager \*pMgr)

This initializes the heap memory to one huge free item.

* Assume the driver has actually allocated the memory for the heap and set StorageManager's pBeginStorage to point to it. The driver also sets the StorageManager's shHeapSize and shMinimumSize.
* Clear out the entire heap with 0 values.
* Set up the entire heap as one free node (with appropriate size) and set the StorageManager's pFreeList to point to it. The end of that free node must also have its size.

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

, SMResult \*psmResult)

This returns a pointer to the user's data in an allocated node.

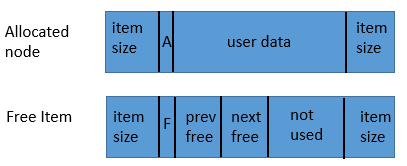
* It is passed the user data size (which is less than the size to allocate due to the node overhead). How much do we allocate? the user data size plus ?
* Although the user may have requested 40 bytes, you must adjust the allocated size by adding the size of the overhead (which contains two item sizes and a one byte flag). See the constant OVERHEAD\_SIZE.
* If the allocated size is less than the StorageManager's minimum size, adjust the allocated size to that value.
* smAlloc 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 psmResult->rc to RC\_NOT\_AVAIL.
  + strcpy an error message to psmResult->szErrorMessage.
  + Functionally return NULL.
* Remove the free item from the free list.
* The remaining portion of the free item might not be big enough for a free item. If not, adjust the allocated size to the original size of the free item.
* If the remaining portion of the free item is large enough for a new free item, insert that at the **front of the free list**.
* Zero out the allocated node prior to copying the data. Copy the data to the allocated node.
* The pointer you return isn't to the beginning of the allocated item, it is to the data portion (after the item size and allocate flag). That is how it works in C when you do a malloc.
* To distinguish allocated items from free items, you will need to set cAF to 'A'.
* Set the size at the front and end of the allocated node.
* If an item is allocated, the SMResult rc should be set to 0. Otherwise, it should be assigned a non-zero value and a message should be returned.

int **smFree**(StorageManager \*pMgr, char \*pUserData, SMResult \*psmResult)

Functionally, this returns the item size of the freed item. The size can be larger than the allocated item's size due to combining with adjacent free items.

* Besides a pointer to the StorageManager, this is passed a user data pointer (see the diagram).
* Verify that the item to be freed is an allocated item (i.e., its cAF is 'A').
* Check for an adjacent free item following it. If there is one, remove it from the free list and combine it with the one being freed.
* Check for an adjacent free item before it. If there is one, remove it from the free list and combine it with the one being freed. It should be easy to check the preceding node since there is a size at the end of the preceding node. To help check the preceding node to see if it is free, you must use the size at the end of that node. Instead, if you traverse the free list to see whether the preceding node is free, you will lose at least 20 points.
* Zero-out the free node before setting its length and inserting it in the free list.
* Insert the new (possibly combined) free item onto the **front of the free list**.
* Return the actual (possibly combined) size of the free item.

Diagrams:



Notes:

1. It is strongly recommended (since modularity matters) that you have additional functions (at least two of them).
2. To simplify grading, please include your C code and output 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**. Your attached output must have been generated on a fox server.
5. For Microsoft Visual Studio Users while doing initial development:

* Since I provided the hexDump as a .o file, you can't use it. Inside the driver, rename dumbHexDump to hexDump. Please remember to delete it 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:**

\* Initialize the Heap

>>> RESET 600

INITIALIZE:

Heap Size: 600

Minimum Node Size: 26

Prefix Node Size: 3

Tot Overhead Size: 5

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x1059070

Minimum Amount is 26

Address Mem Size

01059070 Free 600 End Size= 600

Prev=00000000,Next=00000000

>>> ALLOC 100 FIRST FIRST 100

\* #1 Show the one allocated and one free node

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x10590d9

Minimum Amount is 26

Address Mem Size

01059070 Alloc 105 End Size= 105

01059073 F I R S T 1 0 0 . . . . . . . . . . .

46 49 52 53 54 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059087 thru 010590D6 (4 lines) suppressed - all zero

010590D7 i .

69 00

010590D9 Free 495 End Size= 495

Prev=00000000,Next=00000000

>>> ALLOC 50 SECOND SECOND 50

\* #2 Show the two allocated and the now smaller free node

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x1059110

Minimum Amount is 26

Address Mem Size

01059070 Alloc 105 End Size= 105

01059073 F I R S T 1 0 0 . . . . . . . . . . .

46 49 52 53 54 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059087 thru 010590D6 (4 lines) suppressed - all zero

010590D7 i .

69 00

010590D9 Alloc 55 End Size= 55

010590DC S E C O N D 5 0 . . . . . . . . . . .

53 45 43 4F 4E 44 20 35 30 00 00 00 00 00 00 00 00 00 00 00

010590F0 thru 01059103 (1 lines) suppressed - all zero

01059104 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

01059110 Free 440 End Size= 440

Prev=00000000,Next=00000000

>>> ALLOC 100 THIRD THIRD 100

>>> ALLOC 50 FOURTH FOURTH 50

>>> ALLOC 50 FIFTH FIFTH 50

>>> ALLOC 40 SIXTH SIXTH 40

>>> ALLOC 100 7TH 7TH 100

>>> ALLOC 40 8TH 8TH 40

>>> ALLOC 40 9TH 9TH 40

!!! Memory not allocated

smAllocate rc=901, Requested amount is 45

\* #3 Show all the nodes that were successfully allocated followed by one small free node

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x10592aa

Minimum Amount is 26

Address Mem Size

01059070 Alloc 105 End Size= 105

01059073 F I R S T 1 0 0 . . . . . . . . . . .

46 49 52 53 54 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059087 thru 010590D6 (4 lines) suppressed - all zero

010590D7 i .

69 00

010590D9 Alloc 55 End Size= 55

010590DC S E C O N D 5 0 . . . . . . . . . . .

53 45 43 4F 4E 44 20 35 30 00 00 00 00 00 00 00 00 00 00 00

010590F0 thru 01059103 (1 lines) suppressed - all zero

01059104 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

01059110 Alloc 105 End Size= 105

01059113 T H I R D 1 0 0 . . . . . . . . . . .

54 48 49 52 44 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059127 thru 01059176 (4 lines) suppressed - all zero

01059177 i .

69 00

01059179 Alloc 55 End Size= 55

0105917C F O U R T H 5 0 . . . . . . . . . . .

46 4F 55 52 54 48 20 35 30 00 00 00 00 00 00 00 00 00 00 00

01059190 thru 010591A3 (1 lines) suppressed - all zero

010591A4 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

010591B0 Alloc 55 End Size= 55

010591B3 F I F T H 5 0 . . . . . . . . . . . .

46 49 46 54 48 20 35 30 00 00 00 00 00 00 00 00 00 00 00 00

010591C7 thru 010591DA (1 lines) suppressed - all zero

010591DB . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

010591E7 Alloc 45 End Size= 45

010591EA S I X T H 4 0 . . . . . . . . . . . .

53 49 58 54 48 20 34 30 00 00 00 00 00 00 00 00 00 00 00 00

010591FE thru 01059211 (1 lines) suppressed - all zero

01059212 - .

2D 00

01059214 Alloc 105 End Size= 105

01059217 7 T H 1 0 0 . . . . . . . . . . . . .

37 54 48 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00

0105922B thru 0105927A (4 lines) suppressed - all zero

0105927B i .

69 00

0105927D Alloc 45 End Size= 45

01059280 8 T H 4 0 . . . . . . . . . . . . . .

38 54 48 20 34 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00

01059294 thru 010592A7 (1 lines) suppressed - all zero

010592A8 - .

2D 00

010592AA Free 30 End Size= 30

Prev=00000000,Next=00000000

>>> PRTFREE

Begin Address is 0x1059070

Free List Address is 0x10592aa

Minimum Amount is 26

Address Size Prev Next End Size

010592AA 30 00000000 00000000 30

\*\*\*Total 30

\* Start freeing some nodes

>>> FREE FOURTH

Combined free amount is 55

\* #4 Show the allocated nodes and two free nodes

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x1059179

Minimum Amount is 26

Address Mem Size

01059070 Alloc 105 End Size= 105

01059073 F I R S T 1 0 0 . . . . . . . . . . .

46 49 52 53 54 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059087 thru 010590D6 (4 lines) suppressed - all zero

010590D7 i .

69 00

010590D9 Alloc 55 End Size= 55

010590DC S E C O N D 5 0 . . . . . . . . . . .

53 45 43 4F 4E 44 20 35 30 00 00 00 00 00 00 00 00 00 00 00

010590F0 thru 01059103 (1 lines) suppressed - all zero

01059104 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

01059110 Alloc 105 End Size= 105

01059113 T H I R D 1 0 0 . . . . . . . . . . .

54 48 49 52 44 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059127 thru 01059176 (4 lines) suppressed - all zero

01059177 i .

69 00

01059179 Free 55 End Size= 55

Prev=00000000,Next=010592AA

010591B0 Alloc 55 End Size= 55

010591B3 F I F T H 5 0 . . . . . . . . . . . .

46 49 46 54 48 20 35 30 00 00 00 00 00 00 00 00 00 00 00 00

010591C7 thru 010591DA (1 lines) suppressed - all zero

010591DB . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

010591E7 Alloc 45 End Size= 45

010591EA S I X T H 4 0 . . . . . . . . . . . .

53 49 58 54 48 20 34 30 00 00 00 00 00 00 00 00 00 00 00 00

010591FE thru 01059211 (1 lines) suppressed - all zero

01059212 - .

2D 00

01059214 Alloc 105 End Size= 105

01059217 7 T H 1 0 0 . . . . . . . . . . . . .

37 54 48 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00

0105922B thru 0105927A (4 lines) suppressed - all zero

0105927B i .

69 00

0105927D Alloc 45 End Size= 45

01059280 8 T H 4 0 . . . . . . . . . . . . . .

38 54 48 20 34 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00

01059294 thru 010592A7 (1 lines) suppressed - all zero

010592A8 - .

2D 00

010592AA Free 30 End Size= 30

Prev=01059179,Next=00000000

>>> PRTFREE

Begin Address is 0x1059070

Free List Address is 0x1059179

Minimum Amount is 26

Address Size Prev Next End Size

01059179 55 00000000 010592AA 55

010592AA 30 01059179 00000000 30

\*\*\*Total 85

\* Free Another node

>>> FREE SIXTH

Combined free amount is 45

\* #5 Show the three free nodes and the remaining allocated nodes

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x10591e7

Minimum Amount is 26

Address Mem Size

01059070 Alloc 105 End Size= 105

01059073 F I R S T 1 0 0 . . . . . . . . . . .

46 49 52 53 54 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059087 thru 010590D6 (4 lines) suppressed - all zero

010590D7 i .

69 00

010590D9 Alloc 55 End Size= 55

010590DC S E C O N D 5 0 . . . . . . . . . . .

53 45 43 4F 4E 44 20 35 30 00 00 00 00 00 00 00 00 00 00 00

010590F0 thru 01059103 (1 lines) suppressed - all zero

01059104 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

01059110 Alloc 105 End Size= 105

01059113 T H I R D 1 0 0 . . . . . . . . . . .

54 48 49 52 44 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059127 thru 01059176 (4 lines) suppressed - all zero

01059177 i .

69 00

01059179 Free 55 End Size= 55

Prev=010591E7,Next=010592AA

010591B0 Alloc 55 End Size= 55

010591B3 F I F T H 5 0 . . . . . . . . . . . .

46 49 46 54 48 20 35 30 00 00 00 00 00 00 00 00 00 00 00 00

010591C7 thru 010591DA (1 lines) suppressed - all zero

010591DB . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

010591E7 Free 45 End Size= 45

Prev=00000000,Next=01059179

01059214 Alloc 105 End Size= 105

01059217 7 T H 1 0 0 . . . . . . . . . . . . .

37 54 48 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00

0105922B thru 0105927A (4 lines) suppressed - all zero

0105927B i .

69 00

0105927D Alloc 45 End Size= 45

01059280 8 T H 4 0 . . . . . . . . . . . . . .

38 54 48 20 34 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00

01059294 thru 010592A7 (1 lines) suppressed - all zero

010592A8 - .

2D 00

010592AA Free 30 End Size= 30

Prev=01059179,Next=00000000

>>> PRTFREE

Begin Address is 0x1059070

Free List Address is 0x10591e7

Minimum Amount is 26

Address Size Prev Next End Size

010591E7 45 00000000 01059179 45

01059179 55 010591E7 010592AA 55

010592AA 30 01059179 00000000 30

\*\*\*Total 130

\* Free the first allocated node and then let's use it to allocate a smaller node

>>> FREE FIRST

Combined free amount is 105

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x1059070

Minimum Amount is 26

Address Mem Size

01059070 Free 105 End Size= 105

Prev=00000000,Next=010591E7

010590D9 Alloc 55 End Size= 55

010590DC S E C O N D 5 0 . . . . . . . . . . .

53 45 43 4F 4E 44 20 35 30 00 00 00 00 00 00 00 00 00 00 00

010590F0 thru 01059103 (1 lines) suppressed - all zero

01059104 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

01059110 Alloc 105 End Size= 105

01059113 T H I R D 1 0 0 . . . . . . . . . . .

54 48 49 52 44 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059127 thru 01059176 (4 lines) suppressed - all zero

01059177 i .

69 00

01059179 Free 55 End Size= 55

Prev=010591E7,Next=010592AA

010591B0 Alloc 55 End Size= 55

010591B3 F I F T H 5 0 . . . . . . . . . . . .

46 49 46 54 48 20 35 30 00 00 00 00 00 00 00 00 00 00 00 00

010591C7 thru 010591DA (1 lines) suppressed - all zero

010591DB . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

010591E7 Free 45 End Size= 45

Prev=01059070,Next=01059179

01059214 Alloc 105 End Size= 105

01059217 7 T H 1 0 0 . . . . . . . . . . . . .

37 54 48 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00

0105922B thru 0105927A (4 lines) suppressed - all zero

0105927B i .

69 00

0105927D Alloc 45 End Size= 45

01059280 8 T H 4 0 . . . . . . . . . . . . . .

38 54 48 20 34 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00

01059294 thru 010592A7 (1 lines) suppressed - all zero

010592A8 - .

2D 00

010592AA Free 30 End Size= 30

Prev=01059179,Next=00000000

>>> ALLOC 30 10TH 10TH

\* #6 There should now be some free space after "10TH" and before "SECOND"

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x1059093

Minimum Amount is 26

Address Mem Size

01059070 Alloc 35 End Size= 35

01059073 1 0 T H . . . . . . . . . . . . . . . .

31 30 54 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

01059087 . . . . . . . . . . # .

00 00 00 00 00 00 00 00 00 00 23 00

01059093 Free 70 End Size= 70

Prev=00000000,Next=010591E7

010590D9 Alloc 55 End Size= 55

010590DC S E C O N D 5 0 . . . . . . . . . . .

53 45 43 4F 4E 44 20 35 30 00 00 00 00 00 00 00 00 00 00 00

010590F0 thru 01059103 (1 lines) suppressed - all zero

01059104 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

01059110 Alloc 105 End Size= 105

01059113 T H I R D 1 0 0 . . . . . . . . . . .

54 48 49 52 44 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059127 thru 01059176 (4 lines) suppressed - all zero

01059177 i .

69 00

01059179 Free 55 End Size= 55

Prev=010591E7,Next=010592AA

010591B0 Alloc 55 End Size= 55

010591B3 F I F T H 5 0 . . . . . . . . . . . .

46 49 46 54 48 20 35 30 00 00 00 00 00 00 00 00 00 00 00 00

010591C7 thru 010591DA (1 lines) suppressed - all zero

010591DB . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

010591E7 Free 45 End Size= 45

Prev=01059093,Next=01059179

01059214 Alloc 105 End Size= 105

01059217 7 T H 1 0 0 . . . . . . . . . . . . .

37 54 48 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00

0105922B thru 0105927A (4 lines) suppressed - all zero

0105927B i .

69 00

0105927D Alloc 45 End Size= 45

01059280 8 T H 4 0 . . . . . . . . . . . . . .

38 54 48 20 34 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00

01059294 thru 010592A7 (1 lines) suppressed - all zero

010592A8 - .

2D 00

010592AA Free 30 End Size= 30

Prev=01059179,Next=00000000

>>> PRTFREE

Begin Address is 0x1059070

Free List Address is 0x1059093

Minimum Amount is 26

Address Size Prev Next End Size

01059093 70 00000000 010591E7 70

010591E7 45 01059093 01059179 45

01059179 55 010591E7 010592AA 55

010592AA 30 01059179 00000000 30

\*\*\*Total 200

\* #7 When we Free 8th, it should get combined with the free space that follows it.

>>> FREE 8TH

Combined free amount is 75

>>> PRTALL

Begin Address is 0x1059070

Free List Address is 0x105927d

Minimum Amount is 26

Address Mem Size

01059070 Alloc 35 End Size= 35

01059073 1 0 T H . . . . . . . . . . . . . . . .

31 30 54 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

01059087 . . . . . . . . . . # .

00 00 00 00 00 00 00 00 00 00 23 00

01059093 Free 70 End Size= 70

Prev=0105927D,Next=010591E7

010590D9 Alloc 55 End Size= 55

010590DC S E C O N D 5 0 . . . . . . . . . . .

53 45 43 4F 4E 44 20 35 30 00 00 00 00 00 00 00 00 00 00 00

010590F0 thru 01059103 (1 lines) suppressed - all zero

01059104 . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

01059110 Alloc 105 End Size= 105

01059113 T H I R D 1 0 0 . . . . . . . . . . .

54 48 49 52 44 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00

01059127 thru 01059176 (4 lines) suppressed - all zero

01059177 i .

69 00

01059179 Free 55 End Size= 55

Prev=010591E7,Next=00000000

010591B0 Alloc 55 End Size= 55

010591B3 F I F T H 5 0 . . . . . . . . . . . .

46 49 46 54 48 20 35 30 00 00 00 00 00 00 00 00 00 00 00 00

010591C7 thru 010591DA (1 lines) suppressed - all zero

010591DB . . . . . . . . . . 7 .

00 00 00 00 00 00 00 00 00 00 37 00

010591E7 Free 45 End Size= 45

Prev=01059093,Next=01059179

01059214 Alloc 105 End Size= 105

01059217 7 T H 1 0 0 . . . . . . . . . . . . .

37 54 48 20 31 30 30 00 00 00 00 00 00 00 00 00 00 00 00 00

0105922B thru 0105927A (4 lines) suppressed - all zero

0105927B i .

69 00

0105927D Free 75 End Size= 75

Prev=00000000,Next=01059093

>>> PRTFREE

Begin Address is 0x1059070

Free List Address is 0x105927d

Minimum Amount is 26

Address Size Prev Next End Size

0105927D 75 00000000 01059093 75

01059093 70 0105927D 010591E7 70

010591E7 45 01059093 01059179 45

01059179 55 010591E7 00000000 55

\*\*\*Total 245