ODYSSEUS/EduCOSMOS Q&A

Version 1.0

Advanced Information Technology Research Center (AITrc)
KAIST

ODYSSEUS/EduCOSMOS Q&A

- **EduBfM**

1-1	Title	The key matching and "NOTFOUND_IN_HTABLE"
	Question	When does the function look up return the error message "NOTFOUND_IN_HTABLE"?
	Answer	The look up function returns "NOTFOUND_IN_HTABLE" when the input key does not match any of the registered entries in *hashTable*.

1-2	Title	Memory management in functions
	Question	Q1. Is there any function that requires explicit memory allocation and deallocation using *malloc()* and *free()* for the implementation? Q2. When I test the *setDirty()* function, I get a segmentation fault. I guess this error occurs regardless whether the function is correctly implemented or not. If this is the case, is there any other possible explanation for this fault?
	Answer	A1. Since the memory management is done by *bufferpool*, you don't have to use *malloc()* and *free()* function explicitly. A2. In the test program, *EduBfM_SetDirty()* calls not only *EduBfM_SetDirty()* but also *EduBfM_GetTrain()*. If the implementation of *EduBfM_SetDirty()* is correct, check whether other API functions have been incorrectly implemented.

1-3	Title	Multiple questions
	Question	Q1. What does *nextHashEntry* refer to? Q2. If there is no page having the same value of the hash key, what value should be stored in *nextHashEntry*? Q3. When an error occurs, what kind of actions should be carried out? Q4. What is the role of the macro *HASHTABLESIZE_TO_NBUFS(_x) ((_x)*3-1)?* Q5. What does the value *NextVictim* store? Q6. How do we initialize *bufTable* element? Q7. Is the sentence "the value of the variable *fixed* cannot be less than 0" correct? Q8. In *EduBfM_GetTrain()*, if the page/train does not reside in the buffer pool, what value should be set for the variable *fixed*?
	Answer	A1. *nextHashEntry* refers to *bufTable* index. A2. If there is no page having the same value of the hash key, *nextHashEntry* stores *NIL*. A3. If an error occurs, the log file is to be created and the program terminated. A4. The macro *HASHTABLESIZE_TO_NBUFS(_x) ((_x)*3-1)* returns the size of *hashTable*. A5. *NextVictim* stores the array index of the next buffer element to visit to find the element to replace. A6. The *bufTable* element is initialized iteratively. A7. The statement is correct since the value of the variable *fixed* cannot have a value less than 0. A8. The value of the variable *fixed* should be set with 0 if the page/train does not reside in the buffer pool.

1-4	Title	Printing the output of the test program
	Question	Is the output of the test program printed automatically or do I need to print it explicitly by using *printf()* in *FreeTrain()*?
	Answer	The user should explicitly call *printf()* statement in *FreeTrain()* to print the output

		of the test program.

1-5	Title	*EduBfM_DiscardAll()* and the entry value of *hashTable*
	Question	According to the manual, *EduBfM_DiscardAll()* deletes all entries stored in *hashTable*. Is it possible for the entries to have the same previous values of the index after executing *EduBfM_DiscardAll()*?
	Answer	*EduBfM_DiscardAll()* deletes all entries in *hashTable* by setting the value of entries to NIL. Thus, it is not possible for the entries to have the same previous values of the index after executing *EduBfM_DiscardAll()*.

1-6	Title	Calculating *hashValue*

	Question	Q1. When I caculate the *hashValue*, I use :
		`hashValue = BFM_HASH(key, type)`
		`hashValue -> buftable`
		Is this correct? If I look for an element, can I use *hashTable[hashValue]* to get the array index of *buftable*?
		Q2. When I delete an entry of *hashTable*, do I have to set the value to *NULL* or use *free()* ?

	Answer	A1. You're doing correctly. A2. Set it to NIL.

1-7	Title	The initial key value of *bufTable*
	Question	What is the initial key value of *bufTable*? I tried *NULL*, but it caused an error.
	Answer	Use the macro *SET_NILBFMHASHKEY* as the initial key value.

1-8	Title	About *EduBfM_SetDirty()* and the test program

	Question	Q1. In Test 2_1, I get the segmentation fault error message even though I did not implement *EduBfM_SetDirty*(). It seems that there is a bug in the test program. May I modify the test program by myself?
		Q2. The following two lines of codes for the test program causes error. What is the variable *flags*?
		`apage -> header.flags = i + 1;`
		`printf('The header flags value of pageNo %d is setted '%d'₩n, pageID, apage -> header.flags);`

	Answer	A1. *EduBfM_SetDirty*() returns an error if *EduBfM_GetTrain*() is not implemented correctly. You should not modify the test program. A2. In the test program, the variable *flags* in the page header is used for identifying each page. It is not directly related to the error. The error must be caused by incorrectly implementing *EduBfM_GetTrain*().

1-9	Title	Some questions for the EduBfM project
	Question	Q1. When *EduBfM_DiscardAll*() is called, should all data of *buftable* and *hashtable* be deleted regardless of the value of the variable *fixed*? Q2. I understand that, when *EduBfM_FreeTrain*() is called, data in *buftable* are not deleted even though the value of variable *fixed* is 0. Is this correct? Q3. After conducting Test 3_2, the link information remains in *nextHashEntry*. Should I remove it?
	Answer	A1. Yes. A2. Yes.

		A3. No. You do not need to.

1-10	Title	About the arguments of *RDsM_ReadTrain*()
	Question	I think that the page id of the page pointed by the pointer variable *aTrain* whose value is obtained by calling *RDsM_ReadTrain*() is consistent with the argument *trainId*. However, I found that these values are different. Why are they different?
	Answer	They are garbage data in the BfM module since the BfM module does not manage the internal variables for the page such as page id. The OM module does, so these variables must have meaningful values in the OM module.

1-11	Title	*Util_ErrorLog_Printf*(char* msg, ...)
	Question	I found 3 dots (i.e., "…") in the arguments of *Util_ErrorLog_Printf*(). I don't understand what it means.
	Answe	It is called a variable argument. The variable argument allows C API functions to have a variable number of arguments. The well-known API functions such as *printf*() also use this variable argument. However, note that the EduBfM module does not require using *Util_ErrorLog_Printf*().

1-12	Title	About the error message "LRDS_CommitTransaction failed!!!"
	Question	When do I get the 'LRDS_CommitTransaction failed' error message?
	Answer	You get the 'LRDS_CommitTransaction failed' error message when the LRDS module failed to commit the transaction. However, the error message itself does not give you the detailed reason. I recommend checking the "odysseus_error.log" file for getting the detailed description.

1-13	Title	Some questions about EduBfM
	Question	Q1. According to the description on *edubfm_Delete*() in the manual, 'Delete the array index maintaining the remaining array indexes of the buffer elements storing pages/trains with the same hash key value as a linked list.' Which data structure do I have to modify, *bufTable* or hashTable? Q2. According to the project manual, the value of the variable *fixed* cannot be less than 0. What should I do if it is less than 0? Q3. Should I initialize the *refer bit* to be 1 when calling *EduBfM_GetTrain*()?
	Answer	A1. It depends on which page you delete. If you delete the page whose index is saved in *hashTable*, you should modify *hashTable*. Otherwise, you should modify *bufTable*. A2. You should not let *fixed* value become less than 0. A3. Yes.

- **EduOM**

2-1	Title	The meaning of *extNo*
	Question	What is *extNo*? Can I find *SlottedPage* with *extNo*?
	Answer	A file consists of many pages. The pages are allocated from an extent, which consists of multiple contiguous physical pages, and is the unit of physical space allocation. Use of extents guarantees allocation of pages of a file is physically contiguous. You can access the first *SlottedPage* with *extNo*.

2-2	Title	The description of *available space list* in p.10 of the manual
	Question	Q1. Does 'free space' in p.10 of the manual mean a contiguous free space? Q2. Does '10% of the page' in p.10 of the manual refer to '10% of the total page size' or '10% of the data area size in the page'?
	Answer	A1. No, 'free space' means the sum of contiguous free space and unused space. A2. It means '10% of the data area size in the page'.

2-3	Title	Accessing data in the *SlottedPage* & the meaning of IN/OUT parameter
	Question	Q1. How do we access data stored in *SlottedPage*? Q2. In the parameter description, what does IN / OUT means?
	Answer	A3. Get the pointer from *BfM_getTrain()*, then you can access data by typing the code like '*apage->header.nSlots*'. A4. IN refers to an input parameter and OUT refers to an output parameter.

2-4	Title	The difference between the OUT parameter and the return value
	Question	What is the difference between the OUT parameter and the return value?
	Answer	The OUT parameter returns the result of the function with a pointer variable which is given in the input parameter, i.e., by call-by-reference, and the return value returns the result of the function by using the return structure of the function itself.

2-5	Title	The comparison between the solution file and the output file
	Question	When I compare the solution file with my output file, should their *PageID's* be the same? *fid.serial* in *catObjForFile* is different from that in the solution file.
	Answer	No, as you can see in EduOM_Test.c, the *PageID* printed by the solution file consists of *volNo* and *pageNo*, and they are not in the consecutive order.

2-6	Title	The difference between an *internal function* and an *API function* in the manual
	Question	What is the difference between an *internal function* and an *API function* in the manual?
	Answer	An *API function* can be called from outside of the module while an *internal function* can be called only from the inside of the module.

2-7	Title	The meaning of an *extent*
	Question	What does an *extent* mean?
	Answer	The *extent* means a list of physically contiguous pages. You can access the first page by using the extent number.

2-8	Title	The meaning of an error message & the value of the slot number
	Question	Q1. I got an error message 'LRDS_Dismount failed.' What does it mean? Q2. Can I assume the slot number for the first object as 0, and that for the last object as (*nSlots* - 1)?
	Answer	A1. This error message is caused when you did not call *BfM_FreeTrain()* after *BfM_GetTrain()*. You should call *BfM_FreeTrain()*. A2. Correct

2-9	Title	Questions from the manual
	Question	Q1. What is the offset of an object? Q2. How can I access an object using *ObjectID*? Q3. How are the object and the slot connected?

		Q4. What does *unique* mean? Q5. What is *flag* in *SlottedPage*? Q6. How can I choose *available space list* of the appropriate size?
	Answer	A1. The offset indicates the address distance from the start position of the data area to the given object. A2. You can get the page using *pageNo*, *volNo* and you can access to the object using *slotNo* in *ObjectID*. A3. You can find the position of the object by using the offset value in the slot. A4. *Unique* is used for distinguishing the objects having the same *slotNo* in the page. A5. There are many types in the original ODYSSEUS/COSMOS, but EduCOSMOS uses only the *SlottedPage* type. Thus, *flag* doesn't have any other meaning, but just the type of the page. A6. EduCOSMOS tries to select *available space list* as minimum as possible.

2-10	Title	Question about the *extent* and some variables in the header file
	Question	Q1. What is an extent? Q2. I don't understand why there is *SlottedPage* defined in EduOM_Internal.h and *Page* defined in EduOM_common.h?
	Answer	A1. A file consists of many pages. If these pages are scattered on the disk and not adjacent to each other, reading the data from logically consecutive pages becomes inefficient. Thus, we try to place the pages of a file on the disk in a clustered manner using the extents. An *extent* is a group of physically contiguous pages, and we allocate pages from the *extent*. The *extent number* indicates the page identifier of the first page of the *extent*, and the *extent fill factor* indicates the maximum number of pages allocated from one *extent*. A2. You may ignore them; they are not used in this project.

2-11	Title	Difference between *EduBfM_GetTrain*() and *EduBfM_getNewTrain*()
	Question	I guess that *EduBfM_GetTrain()* and *EduBfM_getNewTrain()* are interchangeable. Then, what's the difference between them?
	Answer	If you allocate a new page, you don't need to read a train from the disk since there is no data in the new page. The difference of the two API functions is whether they need to access the disk or not. *EduBfM_GetTrain*() reads a train from the disk. *EduBfM_getNewTrain*() creates a new train without accessing the disk for efficiency. However, it can be used only when you allocate a new train.

2-12	Title	Question about the value of *nextPage* and *prevPage*
	Question	If a page is the last page of a file, is the value of *nextPage* in the page header *NULL*? If a page is the first page of a file, is the value of *prevPage* in the page header *NULL*?
	Answer	Yes, you can also find it in the result of EduOM_TestSolution.

2-13	Title	Question about the file catalog, *offset*, *moveObject*, and *largeObject.*
	Question	Q1. The file catalog is passed to a function with a type of *ObjectID*. Then, should I use the same way of accessing *ObjectID* for accessing the file catalog? Q2. According to the manual, I have to handle *movedObject* in *ReadObject*. What is *movedObject*? Q3. According to the manual, I have to handle *largeObject* in *ReadObject*. What is *largeObject*?

		Q4. I think that I need *volNo* for accessing *firstPage*. But I cannot find *volNo* since the data type of *firstPage* is *pageNo*. How can I access *firstPage* using the file catalog? Q5. Where is the starting point of *offset*? The starting point of the whole page or the starting point of the data area in the page?
	Answer	A1. Yes, the object of that *ObjectID* is the file catalog. A2. You may ignore it since it is not used in this project. A3. You may ignore it since it is not used in this project. A4. *FileID* in *sm_CatOverlayForData* have *volNo* and you can access *firstPage* using it A5. The starting point of the data area in the page

2-14	Title	Question about the volume, *PhysicalFiledID*, and variables.
	Question	Q1. What is *volume*? Q2. What is *PhysicalFileID*? Is it the ID of the first page? Q3. Do I need to use all the variables in the given files?
	Answer	A1. You may regard it as a disk. A2. We recommend using the *pageID* of the first page as the *PhysicalFileID*. However, this is not a meaningful type so you can replace it with *pageID*. A3. No, you don't.

2-15	Title	The way of calculating the contiguous free area
	Question	How can I calculate the contiguous free area?
	Answer	In EduOM_Internal.h, there is a macro for calculating the contiguous free area.

2-16	Title	Question about the compact area
	Question	I calculated the size of the compacted slot manually. But it is different from the result in the solution. Are there some errors in the solution?
	Answer	The compact area is for only the data area, not for the slot area.

2-17	Title	Questions about *Util_getElementFromPool()* and the relationship between *dlHead* and the new element.
	Question	Q1. In *Util_getElementFromPool(), DeallocListElem *dlElem* is declared. To use it, should I just assign the address of *dlElem* or allocate a new *dealloc list element* using *malloc()*? Q2. What's a relationship between *dlHead* and the new element?
	Answer	A1. Assign the address of *dlElem*. A2. The new element should be the next element of *dlHead*. And, the next element of the new element should be the next element of the original *dlHead*.

2-18	Title	The error of "LRDS_Dismount failed" when using *BfM_GetTrain()*and *BfM_FreeTrain()*
	Question	What is the exact reason of the error of "LRDS_Dismount failed" occurring when I misuse *BfM_GetTrain()*and *BfM_FreeTrain()*?
	Answer	Every *BfM_GetTrain()* must have a matching *BfM_FreeTrain()*. If this matching is not done correctly, you get the error message "LRDS_Dismount failed".

2-19	Title	Question about the data structure related to an object
	Question	In the data structure that represents an object, the size of the character array, which is used to store the data of the object, is defined to be MIN_OBJECT_DATA_SIZE.

		How can I store data whose size is larger than MIN_OBJECT_DATA_SIZE?
	Answer	This data structure is used to store the “pointer” pointing to an object stored in the memory area (such as the data area of the page). Allocating the memory to store an object is independent of this data structure.

- **EduBtM**

3-1	Title	About the key length of the data type *SM_VARSTRING*.
	Question	According to the manual, the length of the key is stored in *klen* in *Btm_LeafEntry*. As I read the code of the test program, however, the length of the key is stored not only in *klen* but also in the first 2 bytes of the memory space of *kval* in the case of SM_VARSTRING. Which one should I use as the length of *kval*?
	Answer	The values of *klen* and the first 2 bytes of the memory space of *kval* are the same. In this project, you’d better use *klen* as the length of *kval*.

3-2	Title	About the stop option, *EQ*/*BOF*/*EOF*
	Question	What should I implement when the stop option is *EQ*/*BOF*/*EOF*
	Answer	This project does not require implementing those stop options.

3-3	Title	When to split the page
	Question	What is the exact condition for splitting a page? Should I split it by checking the free space?
	Answer	Yes. If the size of the object being inserted is bigger than the free space, the page should be split.

3-4	Title	Question about the data structures related to an index entry
	Question	What are the differences among btm_InternalEntry, btm_LeafEntry, InternalItem, and LeafItem? Why are they distinguished?
	Answer	btm_InternalEntry or btm_LeafEntry is used as a pointer to access an internal or leaf entry that is stored in the data area of the page. InternalItem or LeafItem is used as a container of an entry for passing it as an argument to a function. The latter should be distinguished from the former because the latter contains not only the value of the entry but also additional information that is necessary for the function call.