Project1 Report

Haiyang Yun
ID:1233970

1. The basic decision on this project:

Page Organization:

To create a buffer manager, I should decide the page organization at first. The first part of file is a metadata space. The size of this metadata is same as frame size, which is 4096. The space following this metadata is for the true data. And I arrange the first byte of every block as a flag to indicate this block is disposed or not. So, every block “size” is FRAMSIZE+sizeof(char)=4097 actually. But the true size of every block is still 4096 bytes.

File metadata:

As I mentioned at the first part. The metadata is stored in the first 4096 bytes space. This metadata is for storing the last-used block of this file. Since we have functions aim at getting the next block and “this” block of a certain file. It is necessary to have this kind of metadata.

Buffer Pool Structure:

In the BM_init function, I generate a 100 elements

structure array, which name is bufferpool. This array is my buffer pool.

Buffer Pool Replacement Policy:

I use MRU(most recently used) as my replacement policy. Since it is easier to be implemented in C code. I just need a extern variable to store the number of last-used.

2. Challenges in developing.

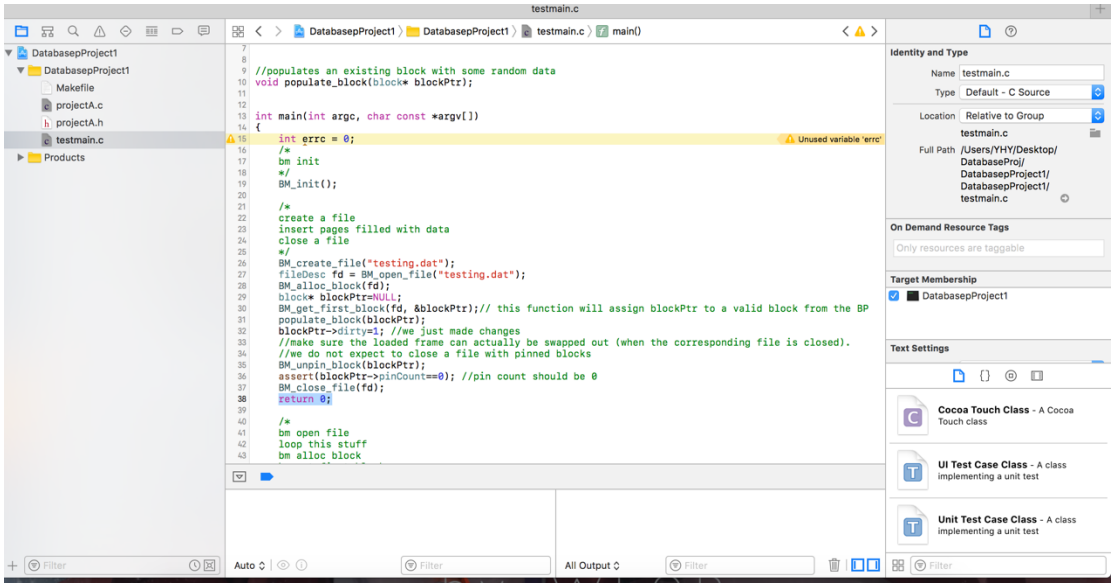
In developing my project code, my challenges mostly caused by unfamiliar with C language and the file organization.

Since this is the first time I use C language to develop a project. I do not know how to use the file pointer to read data from file and put them into my buffer pool. Fortunately, I found some examples in StackOverflow and GitHub. I tried my best to use all these file stream functions though I don’t understand them exactly. But the result should be good.

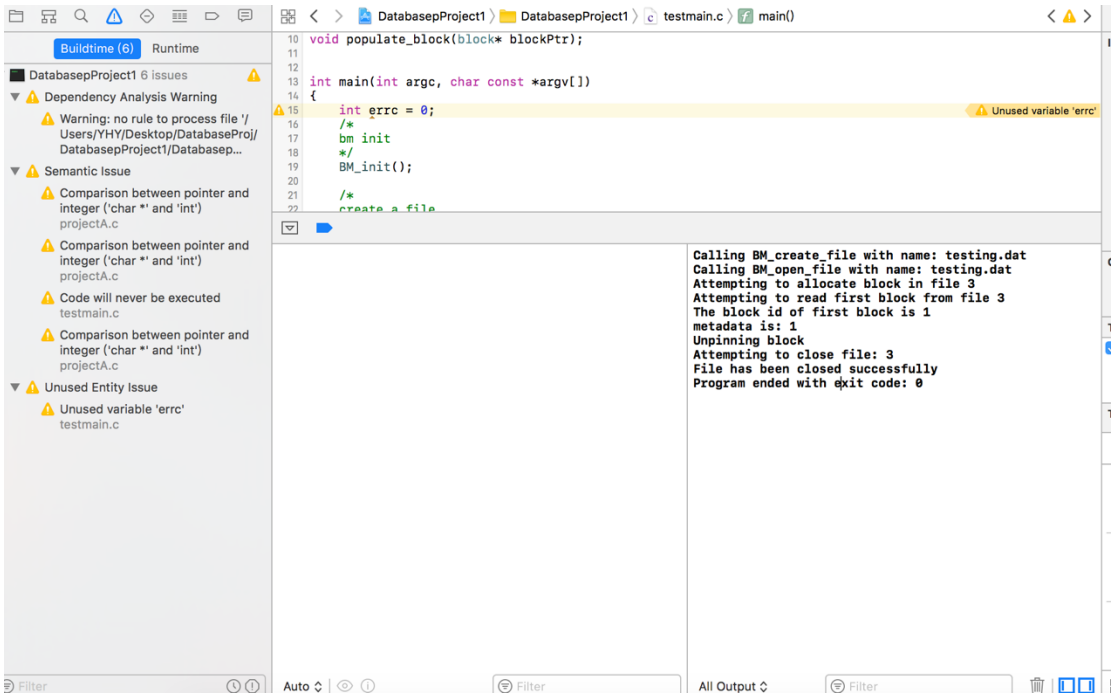
Another challenge is the file organization. Since I do not have an overview of my buffer manager, I have to

change my file's organization when facing different kinds of problem. For example, I didn't define any extern variable at the beginning. It is hard to get the last-used block when implementing MRU policy. So I have to re-define my extern variable.
Next and the last challenge is about the complier. Because I use XCODE in my whole develop. And Xcode has a different compiler with gcc. So, at some circumstance, I can build and compile my testmain in Xcode, but fail compiling in terminal by using gcc(the makefile). And I cannot solve this problem till now.

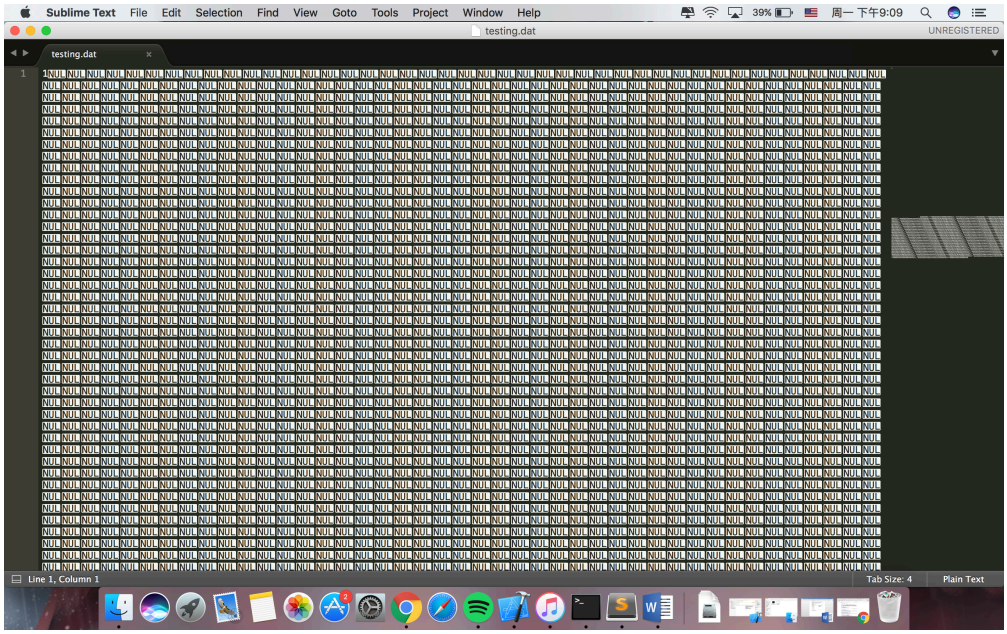
3. The test result.
I use the same testmain as professor provide with. After I finished all my functions, I test my function part by part. At first I give a return 0 at line 38.

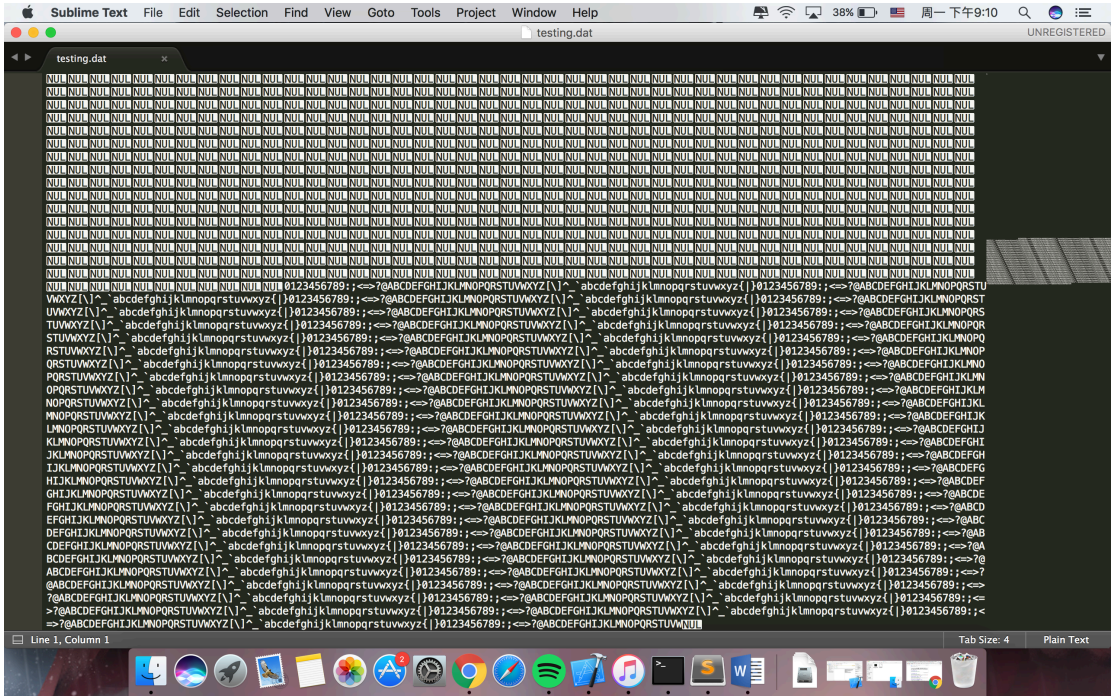


Then I build and run the project. The result is :



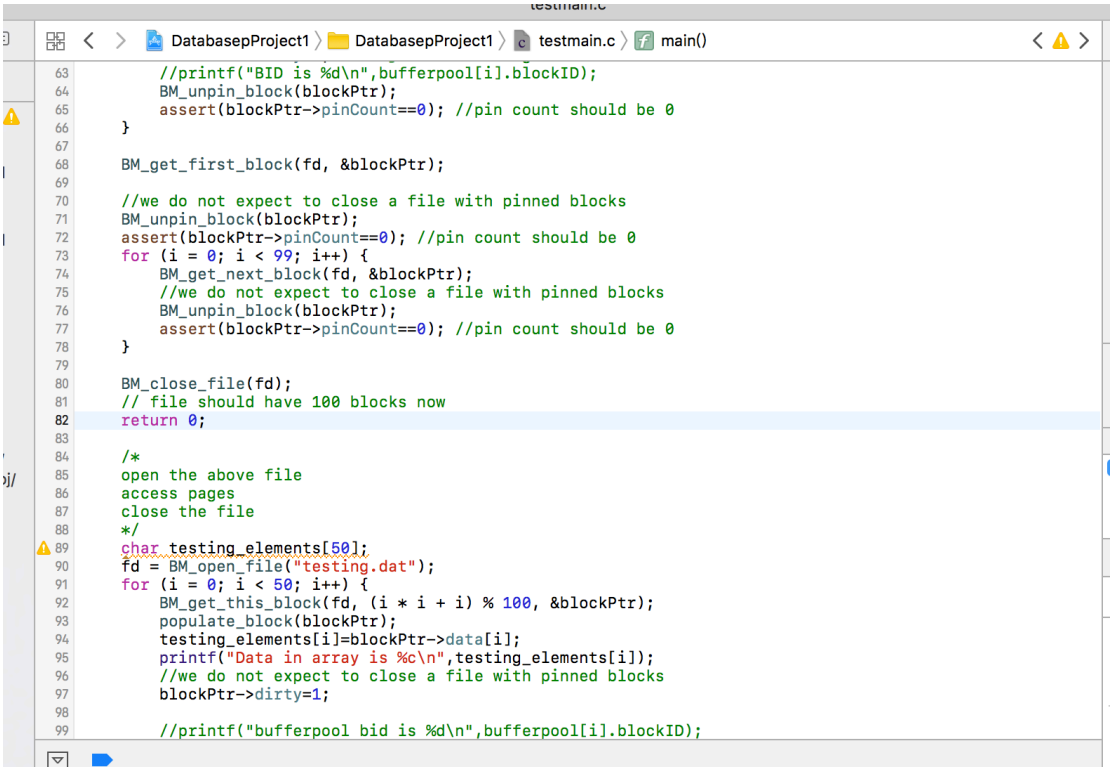
The data in the file is :





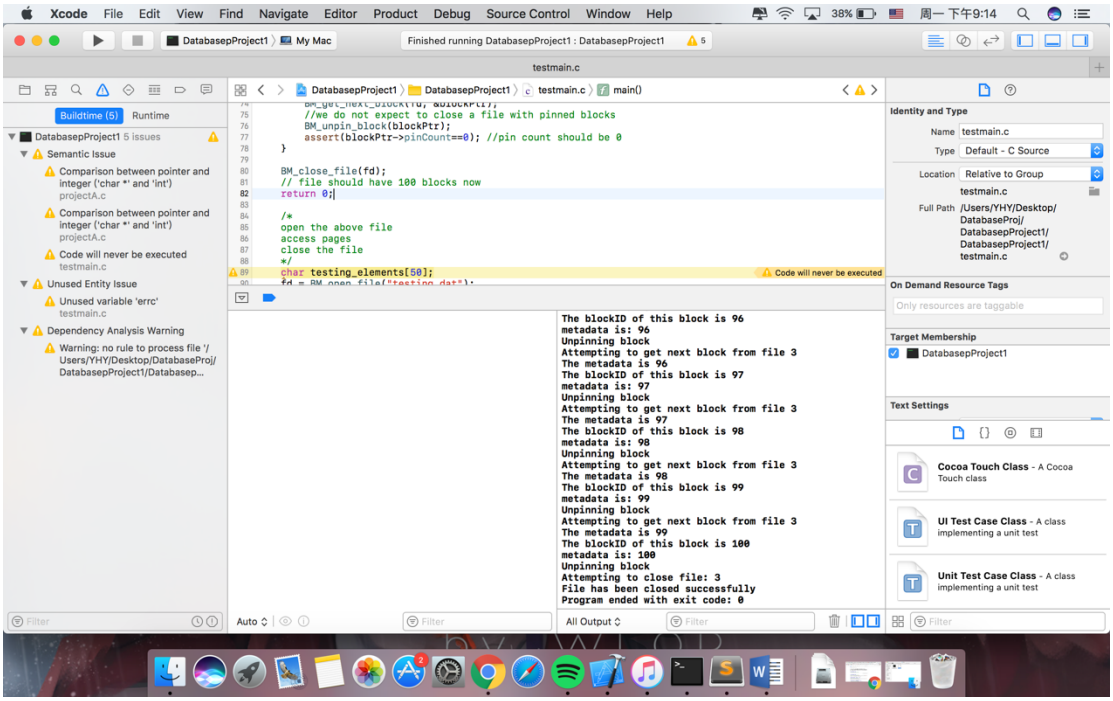
It shows that the block has been put into my file. The file part is metadata with a counting number shows the last-used block. The following part is the data generated by populate_block function.

Then I give return 0 at line 82.



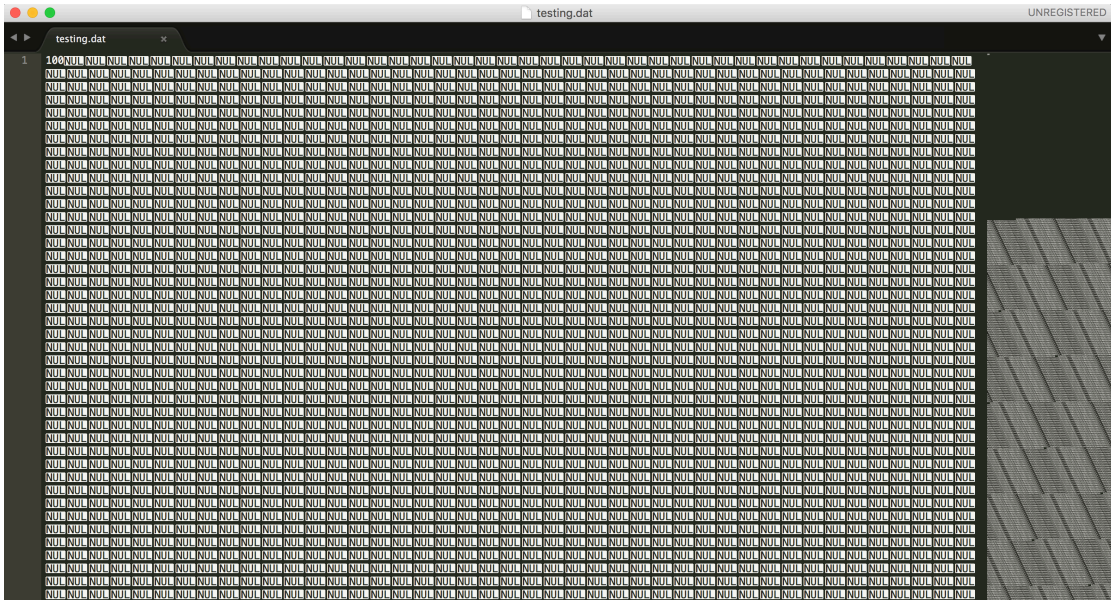
```
63          //printf("BID is %d\n",bufferpool[i].blockID);
64          BM_unpin_block(blockPtr);
65          assert(blockPtr->pinCount==0); //pin count should be 0
66      }
67
68      BM_get_first_block(fd, &blockPtr);
69
70      //we do not expect to close a file with pinned blocks
71      BM_unpin_block(blockPtr);
72      assert(blockPtr->pinCount==0); //pin count should be 0
73      for (i = 0; i < 99; i++) {
74          BM_get_next_block(fd, &blockPtr);
75          //we do not expect to close a file with pinned blocks
76          BM_unpin_block(blockPtr);
77          assert(blockPtr->pinCount==0); //pin count should be 0
78      }
79
80      BM_close_file(fd);
81      // file should have 100 blocks now
82      return 0;
83
84      /*
85      open the above file
86      access pages
87      close the file
88      */
89      char testing_elements[50];
90      fd = BM_open_file("testing.dat");
91      for (i = 0; i < 50; i++) {
92          BM_get_this_block(fd, (i * i + i) % 100, &blockPtr);
93          populate_block(blockPtr);
94          testing_elements[i]=blockPtr->data[i];
95          printf("Data in array is %c\n",testing_elements[i]);
96          //we do not expect to close a file with pinned blocks
97          blockPtr->dirty=1;
98
99          //printf("bufferpool bid is %d\n",bufferpool[i].blockID);
```

The result is



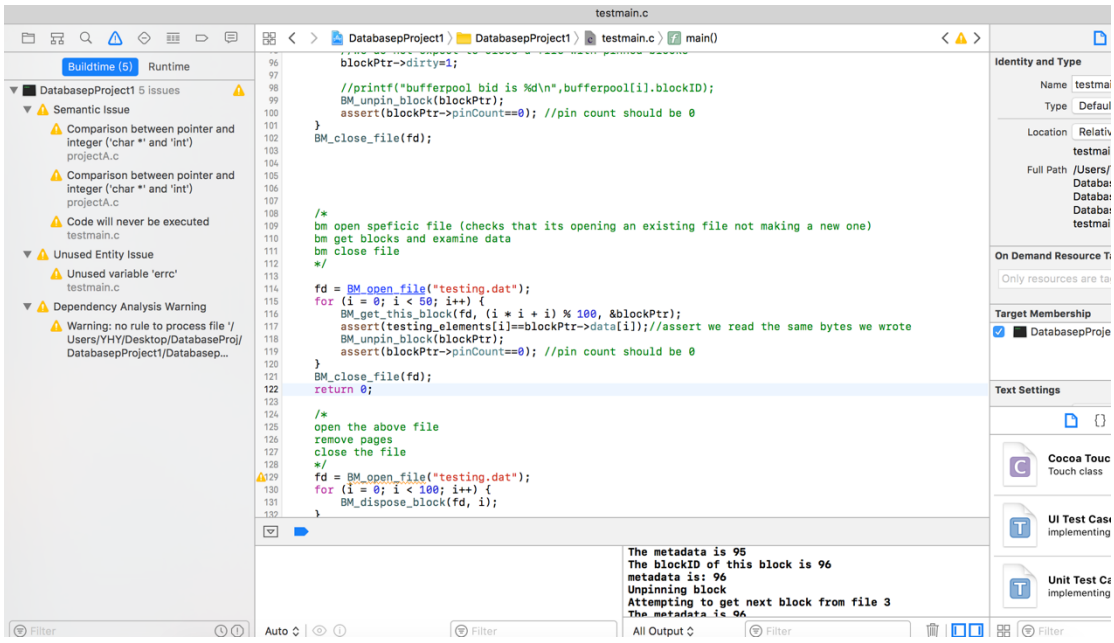
```
The blockID of this block is 96
metadata is: 96
Unpinning block
Attempting to get next block from file 3
The metadata is 96
The blockID of this block is 97
metadata is: 97
Unpinning block
Attempting to get next block from file 3
The metadata is 97
The blockID of this block is 98
metadata is: 98
Unpinning block
Attempting to get next block from file 3
The metadata is 98
The blockID of this block is 99
metadata is: 99
Unpinning block
Attempting to get next block from file 3
The metadata is 99
The blockID of this block is 100
metadata is: 100
Unpinning block
Attempting to close file: 3
File has been closed successfully
Program ended with exit code: 0
```

And the data in file:



The first part is metadata. And all the following space is filled with data.

Then I give a return at line 122:



The result is:

```
    /*
    bm open speficic file (checks that its opening an existing file not making a new one)
    bm get blocks and examine data
    bm close file
    */

    fd = BM_open_file("testing.dat");
    for (i = 0; i < 50; i++) {
        BM_get_this_block(fd, (i * i + i) % 100, &blockPtr);
        assert(testing_elements[i]==blockPtr->data[i]);//assert we read the same bytes we wrote
        BM_unpin_block(blockPtr);
        assert(blockPtr->pinCount==0); //pin count should be 0
```

```
metadata is: 63
Unpinning block
Attempting to get block 56 from file 3
The blockID of this block is 57
metadata is: 57
Unpinning block
Attempting to get block 52 from file 3
The blockID of this block is 53
metadata is: 53
Unpinning block
Attempting to get block 50 from file 3
The blockID of this block is 51
metadata is: 51
Unpinning block
Attempting to close file: 3
File has been closed successfully
Program ended with exit code: 0
```

It means the data we wrote is same as we read. So I pass this part.

At last I run the whole test code. And the result is :

```
    all throughout print some errors if there are any.
    */
    return 0;
}


void populate_block(block* blockPtr) {

    //assumes a block retrieved from BP, updates its contents
    assert(blockPtr!=NULL);

    char* raw_data = malloc((sizeof(char) * FRAMESIZE));
    int i;
    char tmp ;
    for (i = 0; i < FRAMESIZE; i++){
        tmp=i%78+'0';
        raw_data[i] = tmp;
    }
    //here we copy to the contents of the block
    memcpy(blockPtr->data, raw_data, FRAMESIZE);
    free(raw_data);
}
```

```
Attempting to dispose of block 91 from file 3
Attempting to dispose of block 92 from file 3
Attempting to dispose of block 93 from file 3
Attempting to dispose of block 94 from file 3
Attempting to dispose of block 95 from file 3
Attempting to dispose of block 96 from file 3
Attempting to dispose of block 97 from file 3
Attempting to dispose of block 98 from file 3
Attempting to dispose of block 99 from file 3
Attempting to close file: 3
File has been closed successfully
Calling BM_open_file with name: testing.dat
Attempting to read first block from file 3
The block id of first block is 1
metadata is: 1
Unpinning block
Attempting to dispose of block 0 from file 3
Attempting to close file: 3
File has been closed successfully
Program ended with exit code: 0
```

The result of the whole project is correct.

4. More discussion:

The most challenging and interesting part is use pointer to assign data between buffer pool and file. I spend most of my time on doing and debugging this part. And I still make the result perfect.