**Data Structure\_2071035 Lee Somin**

**Technical Report – MemoryAllocation3D.cpp**

*Theorical Explanation of Functions in ‘MemoryAllocation3D.cpp’*

**mem\_alloc\_3D\_double**

inputs: int page, int row, int col

return: double\*\*\* A

Since the program needs to allocate memory for 3-Dementional array, the triple pointer is needed to be assigned. The \*\*\*A of the function ‘mem\_alloc\_3D\_double’ allocates the memory of array of page numbers, and is assigned of the address of the first member [0] of the array of double pointer which contains the address of the rows. Lastly, the single pointer points out to the elements in each row.

**initialize\_3D**

Inputs: double\*\*\* A, int page, int row, int col, int num

Return: non

This function initializes the input 3D array with the given int variable ‘num’ through the ‘for’ statement overlapped 3 times. Each ‘for’ statement represents the page, row and column of the array. To see if the addition function is working correctly, I put the case num==1 to the count++ statement to give variation of the data.

**addition\_3D**

inputs: double\*\*\* A, double B, int page, int row, int col

return: C

3D array A and B are the input of original arrays that are needed to sum up. The function performs the addition through the overlapped ‘for’ statement like the function ‘initialize\_3D’ initialized each element of the array.

When the sizes of twos array are different, it can’t be added, so the function returns NULL and prints out the error message.

**display\_3D**

inputs: double\*\*\* A, int page, int row, int col

return: non

This function prints out each element of given input A, which is 3D array. This function also uses overlapped ‘for’ statement to print out each element and print the line change at every end of row and every end of page.

**deallocate\_3D**

inputs: double\*\*\* A

return: non

This function deallocates 3D array by ‘free’ statement. Since given input is 3D array, it deallocates the column array, row array, and the page array sequentially in a row.

**main()**

input: non

return: non

In main function, the program allocates memory for two 3D array A and B through ‘mem\_alloc\_3D\_double’ and initializes each by the function ‘initialize\_3D’ (To make sure ‘addition\_3D’ is operating correctly, A was initialized as count++ by setting num input as 1.) Then, A and B is added by function ‘addition\_3D’ and the sum is assigned to SumArray. The original array A and B, and the array with sum, SumArray is printed out by ‘display\_3D’ function. Lastly, the memory of the three array is deallocated by the function ‘deallocate\_3D’