```
***********************************************************
*              PROGRAM SPECIFICATION                      *
***********************************************************
```
@PURPOSE:

This program model the effect of different placement algorithms used to assign a process to memory with dynamic partition. In particular, when there are more than one free memory block of the sufficient size, we need to decide which one to allocate. Here, we implement four popular algorithms (first-fit, next-fit, best-fit, worst-fit) and a simulator as well to demonstrate the differences between them.

@CONSTRAINTS:

As this program is to simulate how the four placement algorithms work, there isn't any memory allocation/deallocation operated on real physical memory.

@DESIGN:

1. How to keep track of memory usage:
    1) We use a struct mem_block to record a memory block (either free or used).
    2) All free memory blocks are stored in a std::list called free_blocks (a global variable).
    3) All used memory blocks are stored in a std::list called used_blocks (a global variable).
    4) Memory is initialized as a whole big free block and then splitted into used blocks interleaved with free blocks, as processes are brought in to and swapped out from memory.

2. How to allocate memory (or where to load a process):
    1) To simplify the program, no matter which algorithm is adopted, we search from the beginning of free_blocks until the first one of sufficient size is found.
    2) The only difference between implementation of the four algorithms, in this program, is how to sort the list before searching.

       first-fit: sort free_blocks by start address in ascending order, then do step 2.a (as above).
       next-fit: We use a global variable prev_alloc_block to record the start address of last allocation. Sort free_blocks in ascending start address order, then move all the blocks positioned before prev_alloc_block to the end of the list so that the begining of the list is prev_alloc_block, then do step 2.a.
       best-fit: sort the free blocks by size in ascending order, then do step 2.a.
       worst-fit: sort the free blocks by size in descending order, then do step 2.a.

    3) Once the block is found, we need to remove the original block from free_blocks if the block's size is exactly same as requested. Otherwise, we need to split it into two blocks. The first one is assigned to the process by append it to the end of used_blocks. The info [size and start address] of original block should be modified according to the remaining part.

3. How to free memory (or how to swap out a process):
    1) Search used_blocks until the one allocated to the specified process is found.
    2) Release the memory by moving it back to free_blocks.
    3) We also need to merge those possible contiguous free blocks.

4. Design of simulator:

This simulator first reads a sequence of memory requests (alloc or free) from specified input file and stores them in a vector. Then these requests are processed (with different algorithms) till the first failure. Memory usage is logged in specified output file after each request is processed.

@COMPILER ENVIRONMENT
OS: Microsoft Windows 7 Ultimate
Compiler: Microsoft Visual Studio 2010

```
***********************************************************
*                    HOW TO USE                           *
***********************************************************
```

@CONFIGURATION:
All memory requests are configured in an input file. There are two kinds of requests, either to request allocation or to release allocation. Allocation is requested by "alloc proc_id mem_size" and released by "free proc_id". Following is an example:

```
alloc 1 20
alloc 2 15
free 1
```

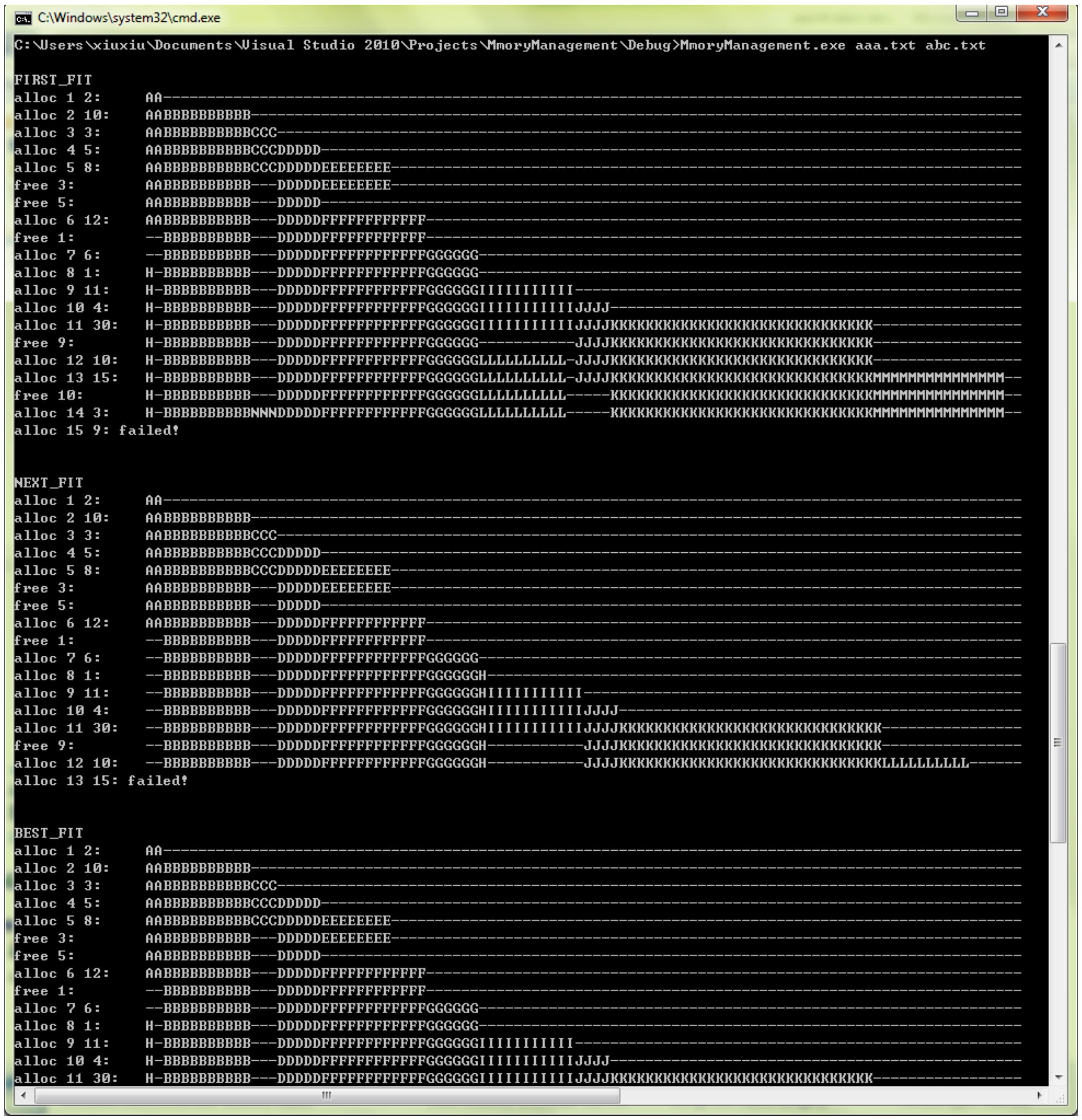
@RUN THE SIMULATOR:
Execute following command in command line: (see figure 1 for run-time effect).

```
       MmoryManagement.exe input_requests_file output_file
```

@HOW TO READ MEMORY USAGE INFO IN OUTPUT FILE:
An simple example of output is as follows:

```
FIRST_FIT
alloc 1 2:      AA-------------------------------------------------------------------------------------------
alloc 2 10:     AABBBBBBBBBB---------------------------------------------------------------------------------
alloc 3 3:      AABBBBBBBBBBCCC------------------------------------------------------------------------------
alloc 4 5:      AABBBBBBBBBBCCCDDDDD-------------------------------------------------------------------------
alloc 5 8:      AABBBBBBBBBBCCCDDDDDEEEEEEEE-----------------------------------------------------------------
free 3:         AABBBBBBBBBB---DDDDDEEEEEEEE-----------------------------------------------------------------
free 5:         AABBBBBBBBBB---DDDDD-------------------------------------------------------------------------
alloc 6 12:     AABBBBBBBBBB---DDDDDFFFFFFFFFFFF-------------------------------------------------------------
```

The first line tells which algorithm is used. In this example, it's first-fit. Then each of the following line is a request followed by a memory usage string. A '-' means the memory unit is free, while a sequence of the same English letters such as 'BBBBBBBBBB' means a block occupied by a process.



```
C:\Users\xiuxiu\Documents\Visual Studio 2010\Projects\MmoryManagement\Debug>MmoryManagement.exe aaa.txt abc.txt

FIRST_FIT
alloc 1 2:     AA--------------------------------------------------------------------------------------------------
alloc 2 10:    AABBBBBBBBBB----------------------------------------------------------------------------------------
alloc 3 3:     AABBBBBBBBBBCCC-------------------------------------------------------------------------------------
alloc 4 5:     AABBBBBBBBBBCCCDDDDD--------------------------------------------------------------------------------
alloc 5 8:     AABBBBBBBBBBCCCDDDDDEEEEEEEE------------------------------------------------------------------------
free 3:        AABBBBBBBBBB---DDDDDEEEEEEEE------------------------------------------------------------------------
free 5:        AABBBBBBBBBB---DDDDD--------------------------------------------------------------------------------
alloc 6 12:    AABBBBBBBBBB---DDDDDFFFFFFFFFFFF--------------------------------------------------------------------
free 1:        --BBBBBBBBBB---DDDDDFFFFFFFFFFFF--------------------------------------------------------------------
alloc 7 6:     --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGG--------------------------------------------------------------
alloc 8 1:     H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGG--------------------------------------------------------------
alloc 9 11:    H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGIIIIIIIIIII---------------------------------------------------
alloc 10 4:    H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGIIIIIIIIIIIJJJJ-----------------------------------------------
alloc 11 30:   H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGIIIIIIIIIIIJJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK-----------------
free 9:        H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGG-----------JJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK-----------------
alloc 12 10:   H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGLLLLLLLLLL-JJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK-----------------
alloc 13 15:   H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGLLLLLLLLLL-JJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKMMMMMMMMMMMMMMM--
free 10:       H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGLLLLLLLLLL-----KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKMMMMMMMMMMMMMMM--
alloc 14 3:    H-BBBBBBBBBBNNNDDDDDFFFFFFFFFFFFGGGGGGLLLLLLLLLL-----KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKMMMMMMMMMMMMMMM--
alloc 15 9: failed!


NEXT_FIT
alloc 1 2:     AA--------------------------------------------------------------------------------------------------
alloc 2 10:    AABBBBBBBBBB----------------------------------------------------------------------------------------
alloc 3 3:     AABBBBBBBBBBCCC-------------------------------------------------------------------------------------
alloc 4 5:     AABBBBBBBBBBCCCDDDDD--------------------------------------------------------------------------------
alloc 5 8:     AABBBBBBBBBBCCCDDDDDEEEEEEEE------------------------------------------------------------------------
free 3:        AABBBBBBBBBB---DDDDDEEEEEEEE------------------------------------------------------------------------
free 5:        AABBBBBBBBBB---DDDDD--------------------------------------------------------------------------------
alloc 6 12:    AABBBBBBBBBB---DDDDDFFFFFFFFFFFF--------------------------------------------------------------------
free 1:        --BBBBBBBBBB---DDDDDFFFFFFFFFFFF--------------------------------------------------------------------
alloc 7 6:     --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGG--------------------------------------------------------------
alloc 8 1:     --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGH-------------------------------------------------------------
alloc 9 11:    --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGHIIIIIIIIIII--------------------------------------------------
alloc 10 4:    --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGHIIIIIIIIIIIJJJJ----------------------------------------------
alloc 11 30:   --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGHIIIIIIIIIIIJJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK----------------
free 9:        --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGH-----------JJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK----------------
alloc 12 10:   --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGH-----------JJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKLLLLLLLLLL------
alloc 13 15: failed!


BEST_FIT
alloc 1 2:     AA--------------------------------------------------------------------------------------------------
alloc 2 10:    AABBBBBBBBBB----------------------------------------------------------------------------------------
alloc 3 3:     AABBBBBBBBBBCCC-------------------------------------------------------------------------------------
alloc 4 5:     AABBBBBBBBBBCCCDDDDD--------------------------------------------------------------------------------
alloc 5 8:     AABBBBBBBBBBCCCDDDDDEEEEEEEE------------------------------------------------------------------------
free 3:        AABBBBBBBBBB---DDDDDEEEEEEEE------------------------------------------------------------------------
free 5:        AABBBBBBBBBB---DDDDD--------------------------------------------------------------------------------
alloc 6 12:    AABBBBBBBBBB---DDDDDFFFFFFFFFFFF--------------------------------------------------------------------
free 1:        --BBBBBBBBBB---DDDDDFFFFFFFFFFFF--------------------------------------------------------------------
alloc 7 6:     --BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGG--------------------------------------------------------------
alloc 8 1:     H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGG--------------------------------------------------------------
alloc 9 11:    H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGIIIIIIIIIII---------------------------------------------------
alloc 10 4:    H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGIIIIIIIIIIIJJJJ-----------------------------------------------
alloc 11 30:   H-BBBBBBBBBB---DDDDDFFFFFFFFFFFFGGGGGGIIIIIIIIIIIJJJJKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK-----------------
```

Figure 1. run the simulator