Princeton University
COS 217: Introduction to Programming Systems
Heap Manager: Algorithms for First Assignment Implementation

void *HeapMgr_malloc(size_t uBytes)

(1) If this is the first call of HeapMgr_malloc(), then initialize the heap manager.

(2) Determine the number of units the new chunk should contain.

(3) For each chunk in the free list...

    If the current free list chunk is big enough...

        If the current free list chunk is close to the requested size, then remove it from the free list, set its status to INUSE, and return it. If the current free list chunk is too big, then remove it from the free list, split the chunk, insert the **tail** end of it into the free list at the front, set the status of the **front** end of it to INUSE, set the status of the **tail** end of it to FREE, and return the **front** end of it.

(4) Ask the OS for more memory — enough for the new chunk. Return NULL if the OS refuses. Create a new chunk using that memory. Insert the new chunk into the free list at the front. If appropriate, coalesce the new chunk and the previous one in memory. To do so, remove the current chunk from the free list, remove the previous chunk in memory from the free list, coalesce them to form a larger chunk, and insert the larger chunk into the free list at the front. Let the current free list chunk be the new chunk.

(5) If the current free list chunk is close to the requested size, then remove it from the free list, set its status to INUSE, and return it. If the current free list chunk is too big, then remove it from the free list, split the chunk, insert the **tail** end of it into the free list at the front, set the status of the **front** end of it to INUSE, set the status of the **tail** end of it to FREE, and return the **front** end of it.

void HeapMgr_free(void *pv)

(1) Set the status of the given chunk to FREE.

(2) Insert the given chunk into the free list at the front.

(3) If appropriate, coalesce the given chunk and the next one in memory. To do so, remove the given chunk from the free list, remove the next chunk in memory from the free list, coalesce them to form a larger chunk, and insert the larger chunk into the free list at the front.

(4) If appropriate, coalesce the given chunk and the previous one in memory. To do so, remove the given chunk from the free list, remove the previous chunk in memory from the free list, coalesce them to form a larger chunk, and insert the larger chunk into the free list at the front.