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

void *HeapMgr_malloc(size_t uiBytes)

```
(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, then...

                If the current free list chunk is close to the requested size, then remove
                it from the free list and return it.  If the current free list chunk is
                too big, then split the chunk and return the tail end of it.  In the
                latter case, the free list need not be altered.

(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 at the end of the free
    list.  If appropriate, coalesce the new chunk and the previous one.  Let the current
    free list chunk be the last one.

(5) If the current free list chunk is close to the requested size, then remove it from the
    free list and return it.  If the current free list chunk is too big, then split the
    chunk and return the tail end of it.  In the latter case, the free list need not be
    altered.
```

void HeapMgr_free(void *pvBytes)

```
(1) Traverse the free list to find the correct spot for the given chunk.

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

(3) If appropriate, coalesce the given chunk and the previous one.

(4) If appropriate, coalesce the given chunk and the next one.
```