Dynamic Memory Allocation Interface

Our generalized interface has two functions:

```
void* allocate_memory(int size);
void deallocate_memory(void* mem_block);
```

Dynamic Memory Allocation Implementation

- **Doubly Linked-Lists:**
  - **Fixed Block Sizes:** Linked-list holds a list of pointers to fixed-size memory locations.
    - **Allocation:** Remove the required number of nodes, return memory addresses in NULL-terminated array.
    - **Deallocation:** Add a new node storing the address of de-allocated location to linked-list.
  - **Variable Block Sizes:** *Each* node represents a *memory location*, *size*, *allocation status*.
    - **Allocation:** Find a node of appropriate size, or split an existing unallocated node.
    - **Deallocation:** Find the allocated node in linked-list; set status to *unallocated.*
- **Bitmaps:** Bit-array keeping track whether each memory block is allocated.

Dynamic Memory Allocation Terms

- **External Fragmentation:** Number of non-coalesced chunks of un-allocated memory.
- **Internal Fragmentation:** Unused memory in an allocated block.
- **Coalesce:** Combining adjacent un-allocated blocks; *reduces* **External Fragementation**
- **Compaction:** Re-positioning allocated memory together to *reduce* **External Fragmentation**

Dynamic Memory Allocation Strategies

Fixed Block Sizes: suitable for embedded systems, where speed is neccessary.

- **One Size Option:** Memory blocks are constant size; multiple (possibly non-contiguous) blocks returned on large allocation request.
- **Varying Size Options:** Fixed-sized memory blocks of different sizes (*generally* powers of 2)

Variable Block Sizes: suitable for general purpose systems.

NOTE: let allocation request be for *n* blocks

- **First Fit** *(Fastest, Best Performance)***:** Find first chunk with *n* unallocated blocks.
- **Next Fit:** essentially **First Fit**, except search for next allocation starts where previous halted.
- **Best Fit:** Find smallest chunk with atleast *n* blocks.
- **Worst Fit:** Find largest chunks with atleast *n* blocks.
- **Quick Fit:** An optimization involving having chunks for commonly used memory allocation requests prepared.
- **Binary Buddy:**
  - **Allocation:** Look for the smallest chunk that has *n* blocks. If *n <= chunk size/2,* then keep splitting the chunk into smaller, and smaller halves.
  - **Deallocation:** After de-allocation, attempt merging chunks of *identical* sizes.