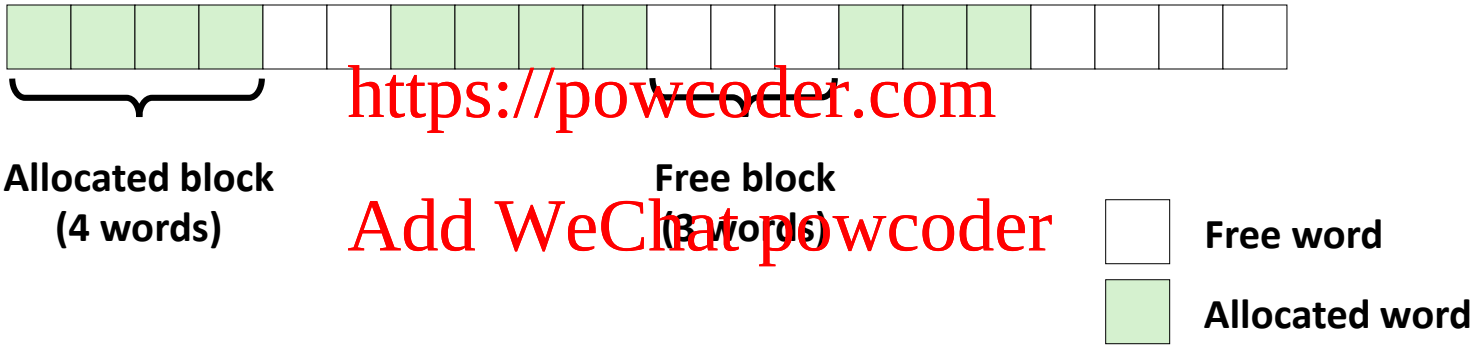
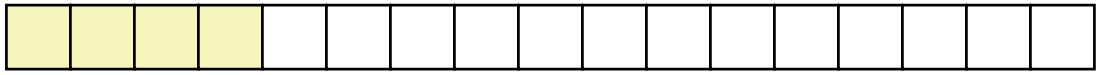
Heap assumptions for lecture

- Memory is word addressed (each word can hold a pointer)



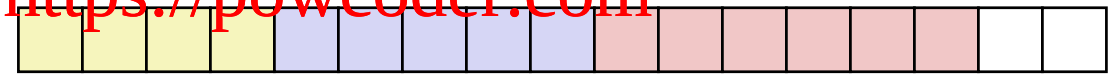
Allocation Example

```
p1 = malloc(4)
```

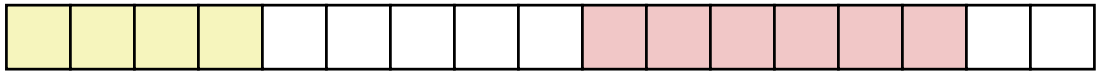


```
p2 = malloc(5)
```

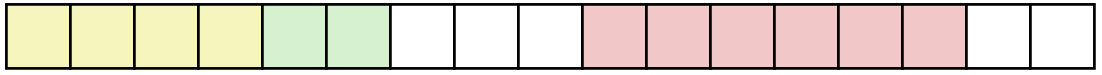

```
p3 = malloc(6)
```




```
free(p2)
```



```
p4 = malloc(2)
```



Constraints

- **Applications**
  - Can issue arbitrary sequence of **`malloc`** and **`free`** requests
  - **`free`** request must be to a **`malloc`**'d block

- **Allocators**
  - Can't control number or size of allocated blocks
  - Must respond immediately to **`malloc`** requests
    - *i.e.*, can't reorder or buffer requests
  - Must allocate blocks from free memory
    - *i.e.*, can only place allocated blocks in free memory
  - Must align blocks so they satisfy all alignment requirements
    - 8 byte alignment for GNU **`malloc`** (**`libc malloc`**) on Linux boxes
  - Can manipulate and modify only free memory
  - Can't move the allocated blocks once they are **`malloc`**'d
    - *i.e.*, compaction is not allowed

Performance Goal: Throughput

- **Given some sequence of `malloc` and `free` requests:**
  - $R_0, R_1, ..., R_k, ..., R_{n-1}$

- **Goals: maximize throughput and peak memory utilization**
  - These goals are often conflicting

- **Throughput:**
  - Number of completed requests per unit time
  - Example:
    - 5,000 **`malloc`** calls and 5,000 **`free`** calls in 10 seconds
    - Throughput is 1,000 operations/second

Performance Goal: Peak Memory Utilization

- **Given some sequence of `malloc` and `free` requests:**
  - $R_0, R_1, ..., R_k, ..., R_{n-1}$
- ***Def:* Aggregate payload P_k**
  - `malloc(p)` results in a block with a *payload* of `p` bytes
  - After request R_k has completed, the ***aggregate payload*** P_k is the sum of currently allocated payloads
- ***Def:* Current heap size H_k**
  - Assume H_k is monotonically nondecreasing
    - i.e., heap only grows when allocator uses `sbrk`
- ***Def:* Peak memory utilization after k requests**
  - $U_k = (max_{i<k} P_i) \ / \ H_k$