# CS 24000 Week 11

Double Pointers and Data Deduplication

# Malloc Recap

- Finds the first available heap space, claims it, and returns its value
  - Technically gives more memory than you ask for
  - Why?
    - “Book-keeping,” or additional data about the memory usage
    - Padding, where data in C is padded out to 4 or 8 byte sections
- Because of this, we try to keep data duplication to a minimum
- **Data deduplication** is the paradigm* of only allocating as much memory as you need, and only storing data in one single place, reusing its memory address

*Paradigm: A philosophy in algorithm design used to solve a problem

# Real-World Deduplication Example

- A common practice in database management is to store data once and **reference** it multiple times
    - Ex: Steam has one database for users, and one database for games
        - Users can play the same games, but those games only need to be stored in one place
    - Major benefit: Update a game once, it updates for everybody

| **Title** | **GameID** |
|---|---|
| The Witcher | 0x123 |
| Crisis | 0x124 |
| Portal | 0x125 |
| ... | ... |

| **Username** | **UserID** | **LibraryID** |
|---|---|---|
| hobsonn | 0x1 | 0x12 |
| gaben | 0x2 | 0x13 |

| **Library 0x12** | **Library 0x12** |
|---|---|
| 0x123 | 0x123 |
| 0x124 | 0x124 |
|  | 0x125 |

Note: This is overly simplified

# HW10 - Using double pointers for Data Deduplication

- In general, you will only be storing “data” once, and referencing it in multiple entries in your “database”
    - This corresponds to “entries” and “transactions” in HW10
- You will only be calling malloc in two places:
    - 1) create_transaction, to actually create the account transaction (entry)
    - 2) get_over_amount_list, to create a new transaction list, reusing the original entries
- The same rules apply: Only malloc as much memory as you need, and always have as many mallocs as frees