Lab Goals

• Work individually or in teams of 2 to apply knowledge of Pointers and Dynamic Allocation to complete 1-3 coding challenges
• Provide time to work on Project 2 and get help from the TAs
Exercise 1: Kinematics

- Modeling the motion of an object in 1 dimension
- Quick physics review:
  - Position = where you’re located
  - Velocity = how quickly you’re moving and in what direction
Exercise 1: Kinematics

• Your program is responsible for keeping track of an object’s position and velocity and the current time.

\[ t = 7 \text{ (units of time)} \]
\[ v = 3 \text{ (units of distance/unit of time)} \]
Exercise 1: Kinematics

Users should be able to…

- **elapse time** - The user says that X units of time pass. Update the current time and the object’s position.

- **cover distance** - The user says that the object travels X distance. Update the current time and the object’s position.

- **get time to position** - Tell the user how long it will take the object to reach a specified position.
Exercise 1: Kinematics

Example 1
> set-position 5
> set-velocity 10
> elapse-time 1
> get-stats
Position: 15
Velocity: 10
Time: 1

Example 2
> set-position 5
> set-velocity 10
> cover-distance 100
> get-stats
Position: 105
Velocity: 10
Time: 10
void elapsedTime(
    double  *position,
    double  velocity,
    double  *time,
    double  timeChange)
Exercise 2: To-Do

Goals:

Build a todo list application
Practice dynamic memory allocation and de-allocation
Gain familiarity with pointers to pointers
Exercise 2: To-Do

Users should be able to…

○ **add an item** - Add a new item to the end of the to-do list.

○ **remove an item** - Remove the last item from the to-do list.

○ **remove all items** - Empty the to-do list.

○ **print** - Print out each item in the to-do list.
To-Do Memory Management

Storage demands:
- Up to 1,000 items in the to-do list
- Each item in the to-do list may consist of up to 200 characters

Memory Allocation:
- You can immediately allocate space for 1,000 pointers to to-do list items
- You should dynamically allocate and de-allocate the to-do list items as they are added and removed
Data Type Considerations

• What data type do you use to store the address of a character array?

• What data type would you use to store the address of an array of character arrays?
Exercise 2: To-Do

Example 1
> add buy-groceries
> add finish-pr2
> add practice-guitar
> remove-last
> print
Todo List:
buy-groceries
finish-pr2

Example 2
> add buy-groceries
> add finish-pr2
> remove-all
> add practice-guitar
> print
Todo List:
practice-guitar
Exercise 3: Patients

Goals:

- Build a patient data management application
- Practice dynamic memory allocation and de-allocation
- Gain familiarity with pointers to pointers
- Be exposed to different data structures
Exercise 3: Patients

Premise:

- Implement a database to keep track of numerical info about hospital patients, like height or blood pressure.

- Compute percentiles based on values in database.

- Program should work for hospitals of different sizes, such as a local urgent care clinic vs the Keck network.
Exercise 3: Patients

Users should be able to...

○ **add a patient** - Store a new patient datum.

○ **remove a patient** - Remove the most recently added patient datum.

○ **compute a percentile** - Determine what proportion of the stored data are less than or equal to a provided value.

○ **print** - Print out each stored patient datum.
Patients Memory Management

Memory Allocation:

- You should begin with an array with space for just 1 patient datum

- When the patient array is full and there is a new datum to be added, double the size of the patient array.

- When a patient is removed and the patient array becomes at least 75% empty, halve the size of the patient array.

- Be sure to appropriately allocate and de-allocate memory
Schedule and Checkoff

First hour:
* To get credit: Complete the 1st exercise (*physics*) and start the second exercise (*todo*). Feel free to work on *patients* as a 3rd exercise.

Second hour:
* Work on PR2 with TAs available