# Fast Food Order Queue

CS 211 – Project 4

COMPUTER SCIENCE
COLLEGE OF ENGINEERING

UIC

## Fast Food Order Queue

- No reservations taken
- Show up and give your order to the cashier
- When an order is prepared, it will be placed on the counter for pick up and the person/group that has been waiting the longest and matches the order ( number of prepared food items is greater or equal to their request) can pick up.

# Fast Food Order Queue

- You can also “Call Ahead” and put your order in.
- Once a Call Ahead party arrives, they must check in and tell the cashier they have arrived

So NOW:

- When an order is available for pick up:
  - the group/person that has been waiting the longest AND
  - their order matches the prepared food items AND
  - is IN the restaurant)
- can pick up that order –> (removed from the Order Queue)

# Fast Food Order Queue

- Implement as a Linked List using the C Language
- Node should contain
  - Name (C style string)
  - Number of burgers ordered (integer)
  - Number of salads ordered (integer)
  - In-Restaurant Status (create Enumerated type – like Boolean)
  - Next Pointer

- Add new node to end of list when an order is added in the restaurant or calls in
- Serve/Remove the node closest to the front that meets conditions

## Fast Food Order Queue

- The restaurant has only one staff in the kitchen at anytime working on preparing ordered food.

- The kitchen staff will only work on one order at a time and on one item at a time, so if an order of 2 burgers and 2 salads comes in, it will take him/her 14 minutes to prepare that order.

- A customer can get an estimated waiting time by asking the cashier. The cashier will give them an estimated time **based on the order of people ahead of them**.

## Fast Food Order Queue Commands

- User Interface should be in base code file (except debug mode stuff)
- Add command: a <#burgers> <#salads> <name>
    - Add to list –order is added in restaurant
    - Validate that name is unique (not already in order queue)
- Call Ahead command: c <#burgers> <#salads> <name>
    - Add to list – group is NOT in restaurant
    - Validate that name is unique (not already in order queue)
- Waiting Command: w <name>
    - Call Ahead order finally arrives in restaurant
    - Change value in order node to indicate they are IN restaurant

# Fast Food Order Queue Commands

- Retrieve command: r <#burgers> <#salads>
  - Remove node from list that matches the prepared order and is IN restaurant
  - Display message if there is no match for removal/pickup
- List command: l <name>
  - List information for all orders ahead of the indicated order name
- Display Command: d
  - Show information for all orders in the list
- Quit Command: q
- Help Command: ?
  - Both should already be programmed for you in base code

## Development Requirements

- Must have (at least) 3 source code “.c” files
  - PLUS:  header “.h” file and makefile
- Function decomposition is Specified
  - Which functions go into which source code files
- Names of Linked Lists Functions are Specified
  - You can create additional functions using whatever name you desire
- Often Program Specs that you MUST follow are given to you

## Development Requirements: Source Code Files

The following functions from proj4base.c are to be in one source code file (these are the user interface functions):

- main()
- clearToEoln()
- getNextNWSChar()
- getPosInt()
- getName()
- printCommands()

## Development Requirements: Source Code Files

The following functions from proj4base.c are to be in another source code file (these are the application functions that interact with the linked list functions): Note Parameters can be changed as needed!

- doAdd()
- doCallAhead()
- doWaiting()
- doRetrieve()
- doList()
- doDisplay()
- doEstimateTime()

# Development Requirements: Source Code Files

- The third source code file is to have the code that you are writing that will perform the linked list implementation.

  The functions in this source code file will include the following functions **plus** any other you write to handle the linked list:

  Given names **MUST** be used, you determine parameters

  - addToList()
  - doesNameExist()
  - updateStatus()
  - retrieveAndRemove()
  - countOrdersAhead()
  - displayOrdersAhead()
  - displayListInformation()
  - displayWaitingTime()

## Development Requirements

- Must have (at least) 3 source code “.c” files
- DON’T FORGET:  header “.h” file and makefile
- Name your executable Project4
- Zip all files together in a folder called “Project4” and submit 1 .zip file to Gradescope

## Running the Program

- Assume the user (the cashier) knows:
    - how to properly enter commands

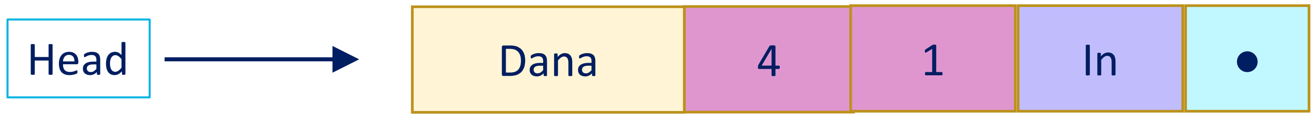
- Assume the user will enter commands when:
    - An order is given in the restaurant
    - Someone calls the restaurant
    - An order is ready for pick up
    - Someone wants to know their estimated wait time
    - etc

# Running the Program

- Assume there is no order in the queue when the program starts
  - The order queue (linked list) is empty

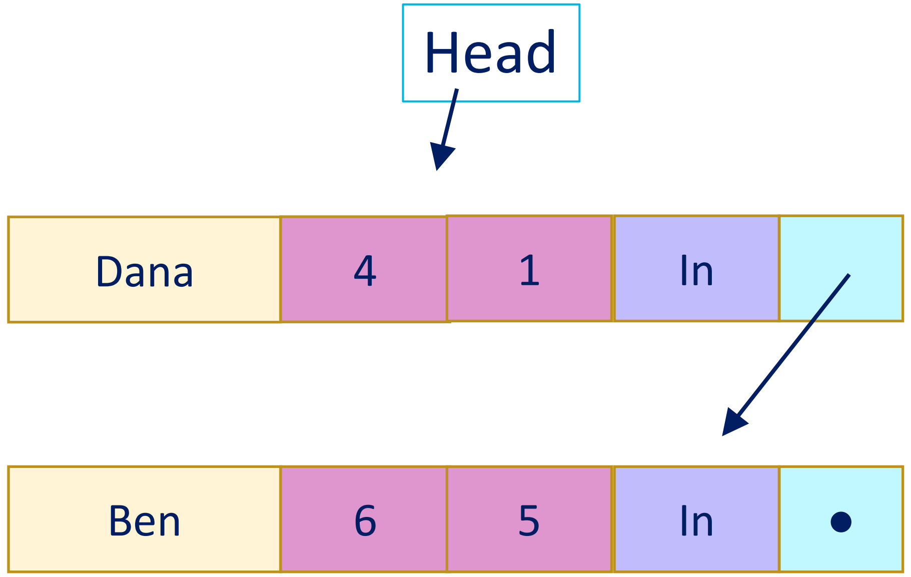
- Once an order is received, it will be added to the order queue:
- Ex: An order of 4 burgers and 1 salad with name of Dana is received in the restaurant
  - Command: a 4 1 Dana

Linked List should look as follows after proper code is run:



## Running the Program

- Now, Ben enters and orders 6 burgers and 5 salads
- Command:   a   6   5  Ben
    - What are the steps?
- Check that name Ben is not in list
- Create new node and add to end of list

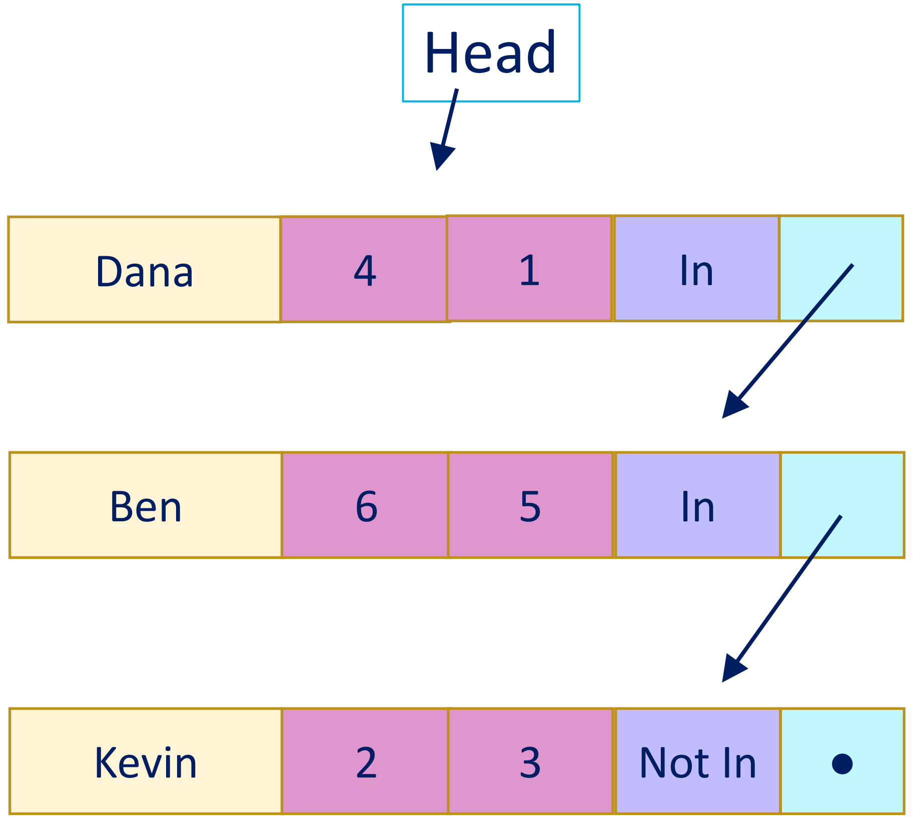


# Running the Program

- Now, Kevin calls ahead and puts an order for 2 burgers and 3 salads
- Command:   c   2   3    Kevin

- Check that name Kevin is not in list

- Create new node and add to end of list

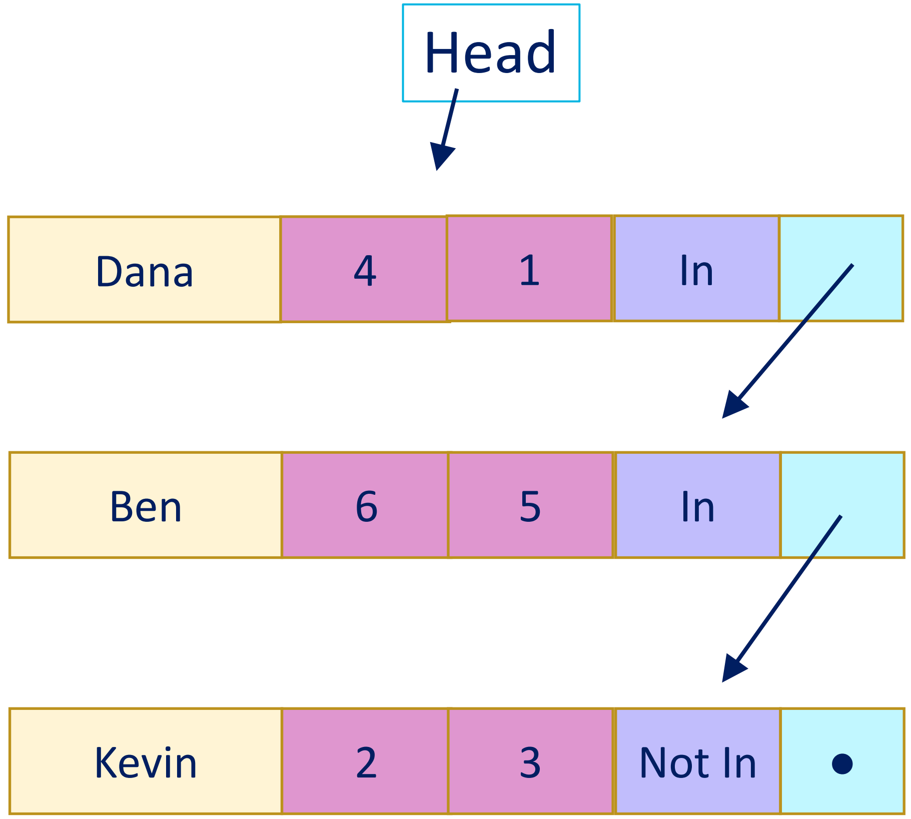


# Running the Program

- Now, someone enters and tries to order 2 burgers and 1 salad under name of Ben,
- Command: a 2 1 Ben

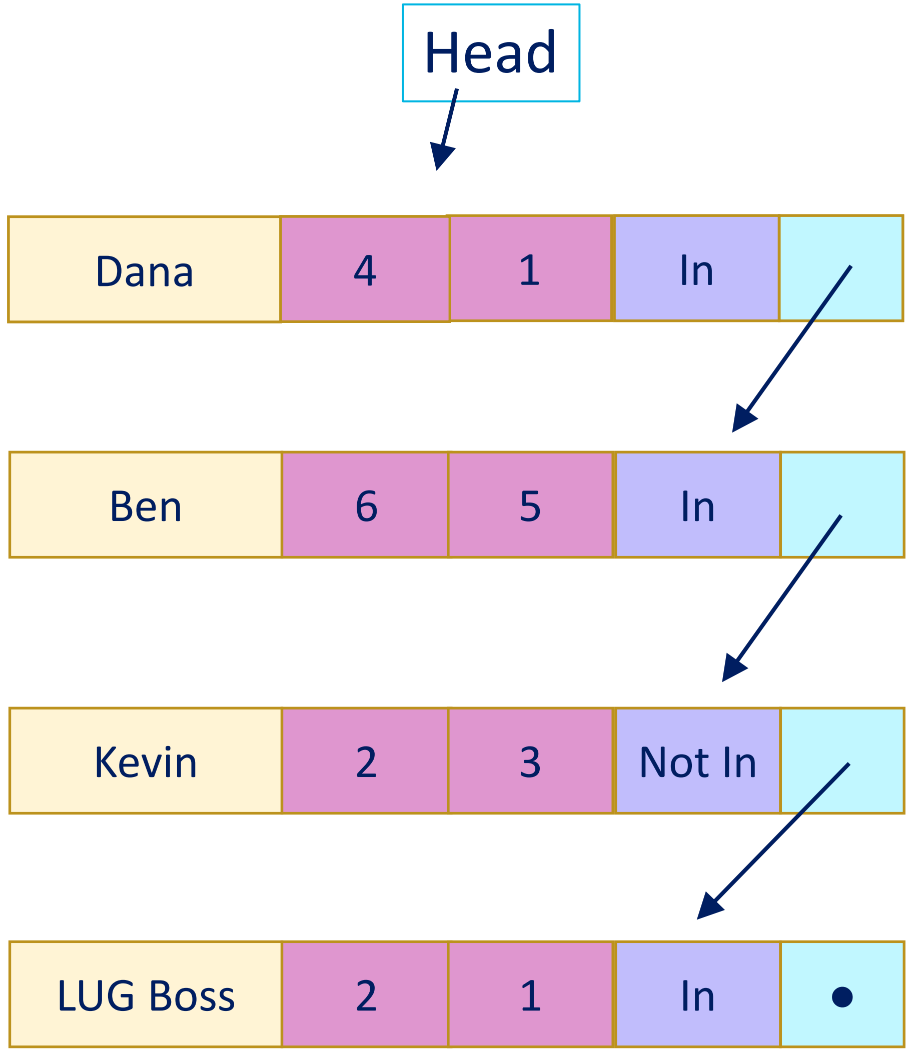
- Check that name Ben is not in list
  - The name already exists!

- Report that name already exists
- Do NOT change list



## Running the Program

- Ben now gives name as “LUG Boss”,
- Command: a 2 1 LUG Boss
- Check that name “LUG Boss” is not in list
- Add new node to end of the list

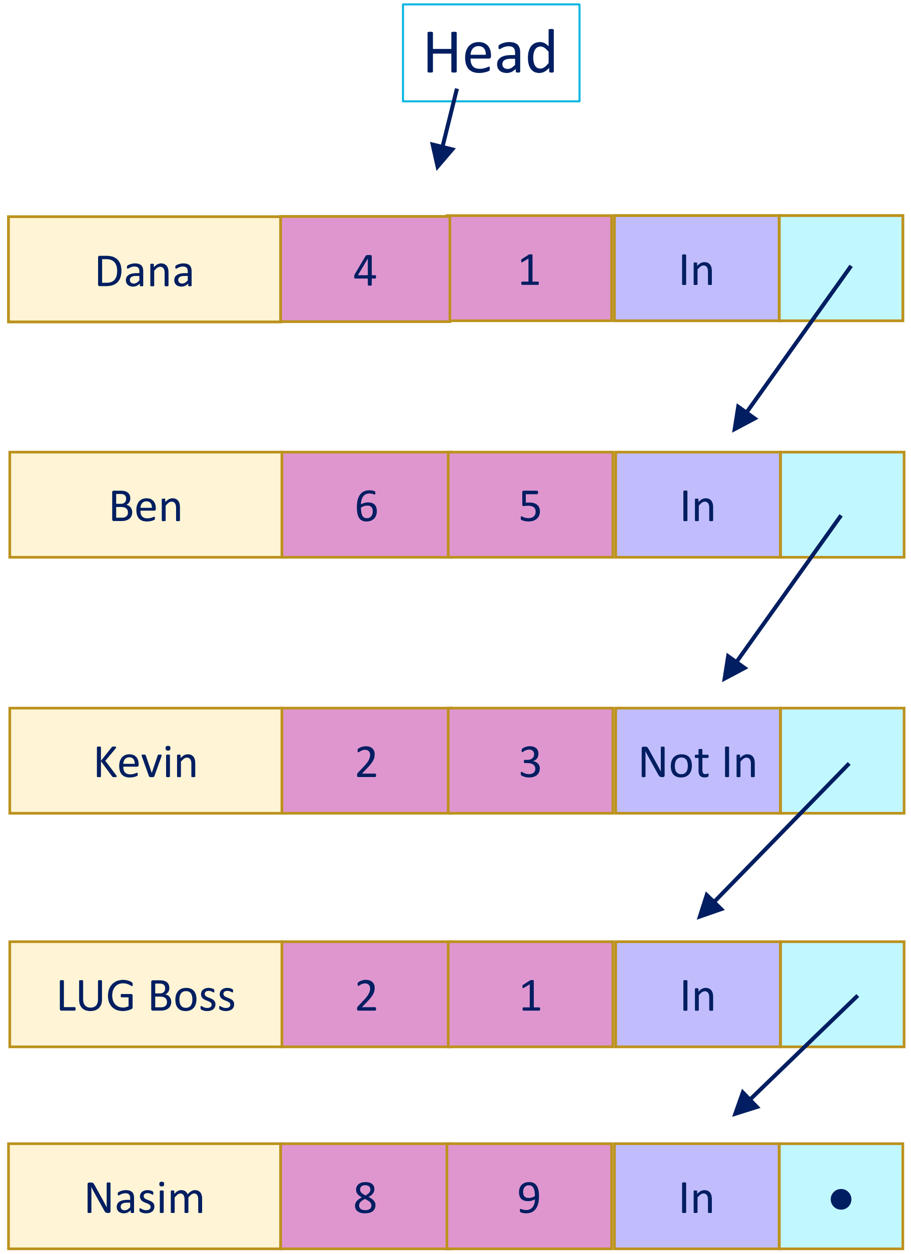


# Running the Program

- Now, Nasim enters and puts an order of 8 burgers and 9 salads
- Command: a 8 9 Nasim

- Check that name Nasim is not in list

- Add new node to end of the list



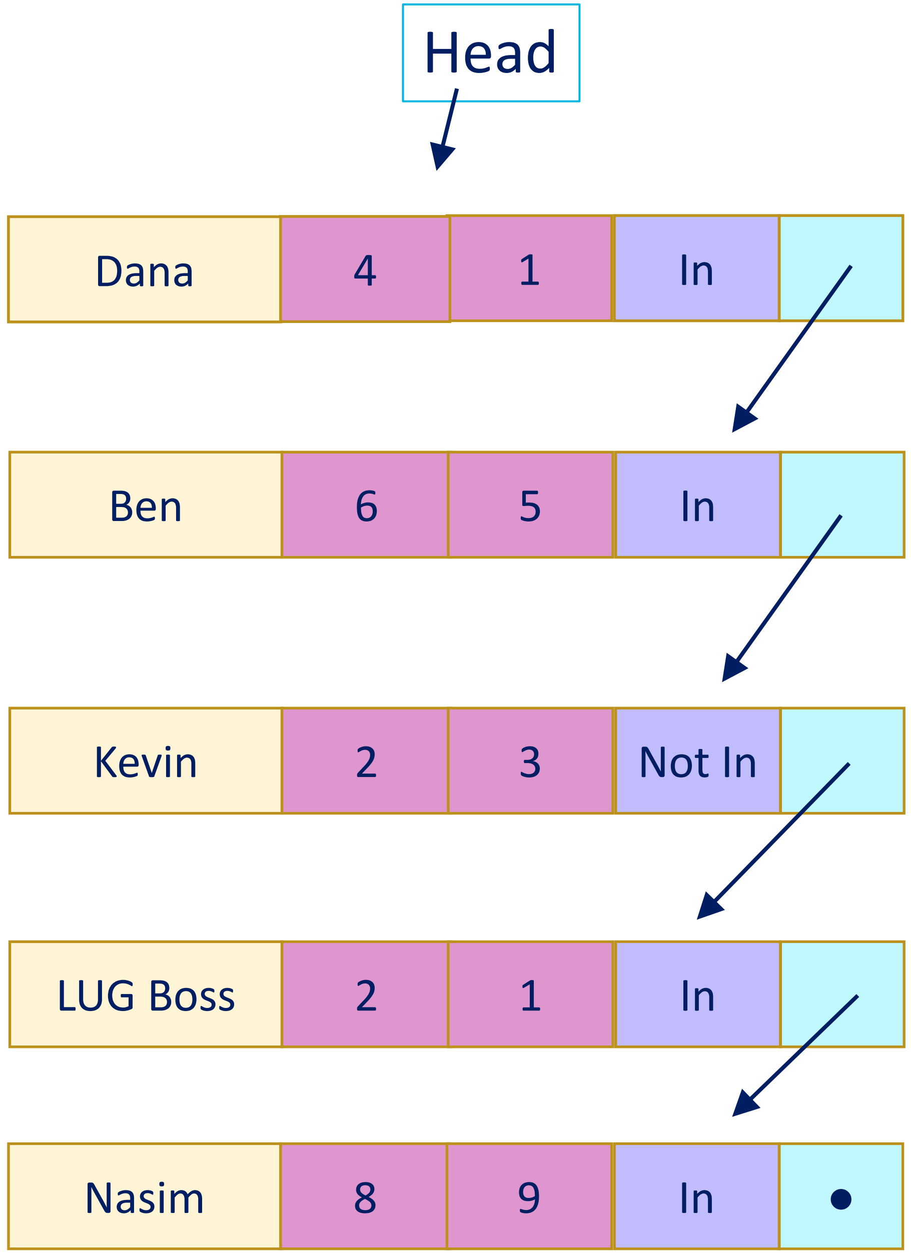
# Running the Program

- LUG Boss become impatient and wants to know how many orders are ahead
- Command:   l   LUG Boss

- However user types name wrong:
- Typed command:   l    lugboss

- Name of “lugboss” is not in list

- Report error that name doesn’t exist



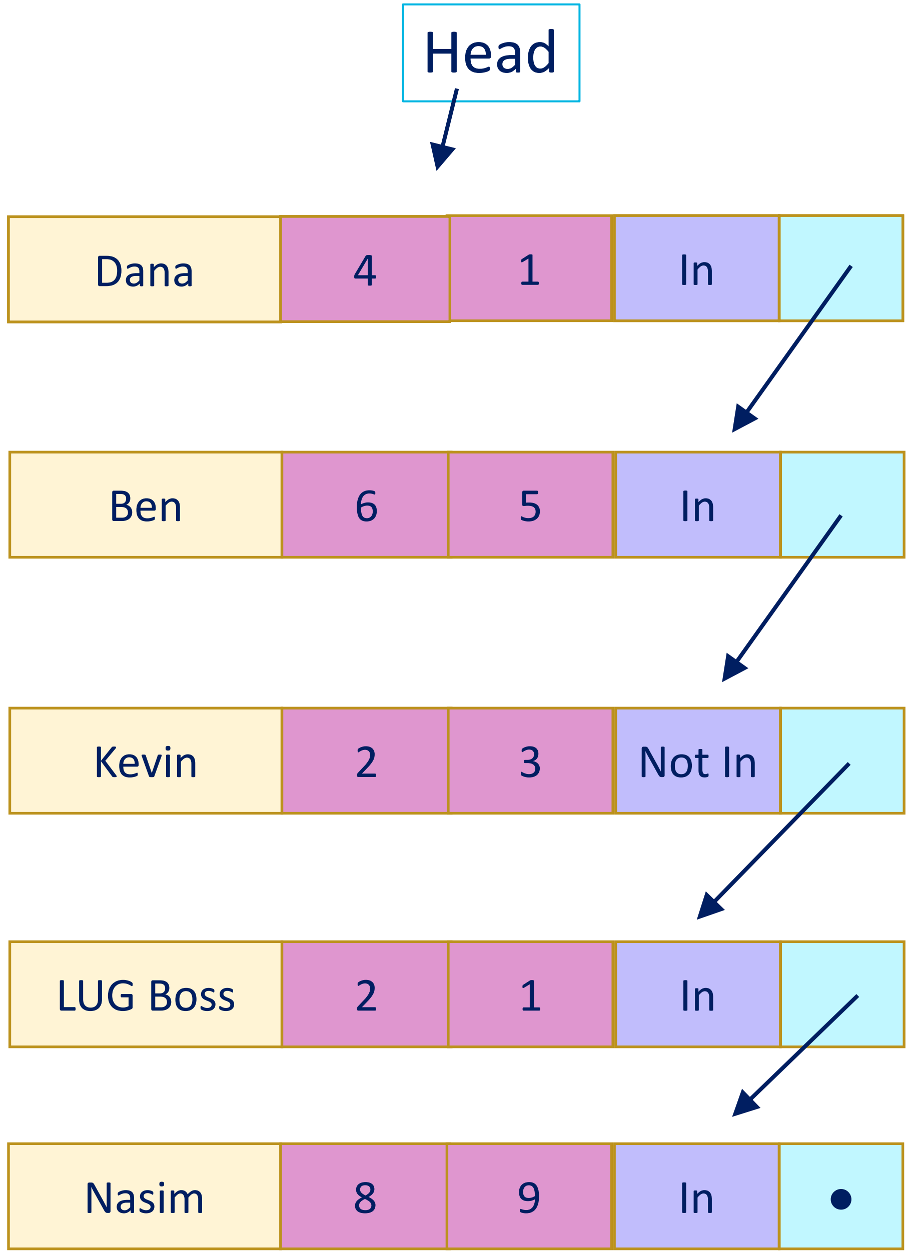
# Running the Program

- LUG Boss become impatient and wants to know how many orders are ahead
- Command:   l   LUG Boss

- Now command is typed correctly, result:

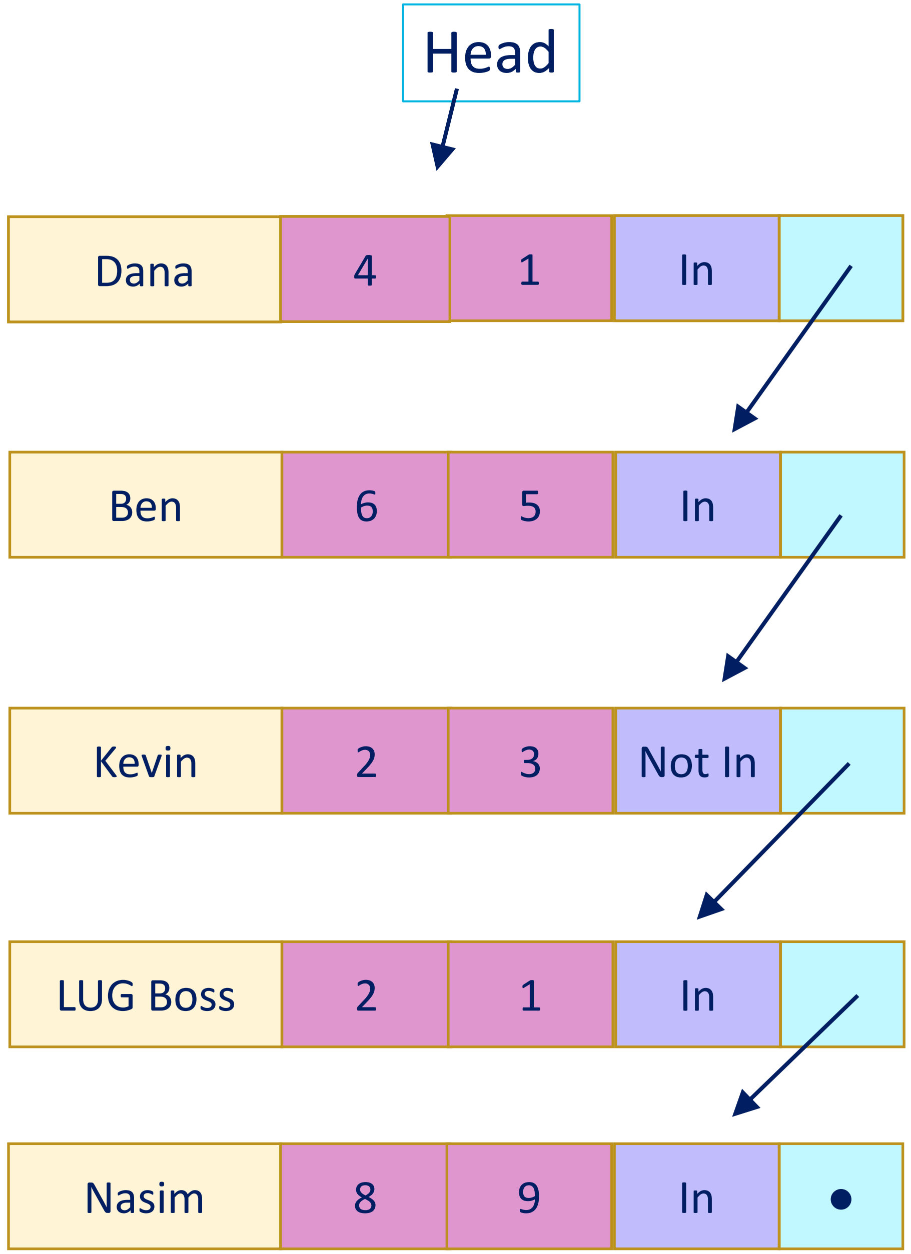
There are three orders ahead of you.
Orders details are:

- Dana: 4 burgers and 1 salad
- Ben: 6 burgers and 5 salads
- Kevin: 2 burgers and 3 salads



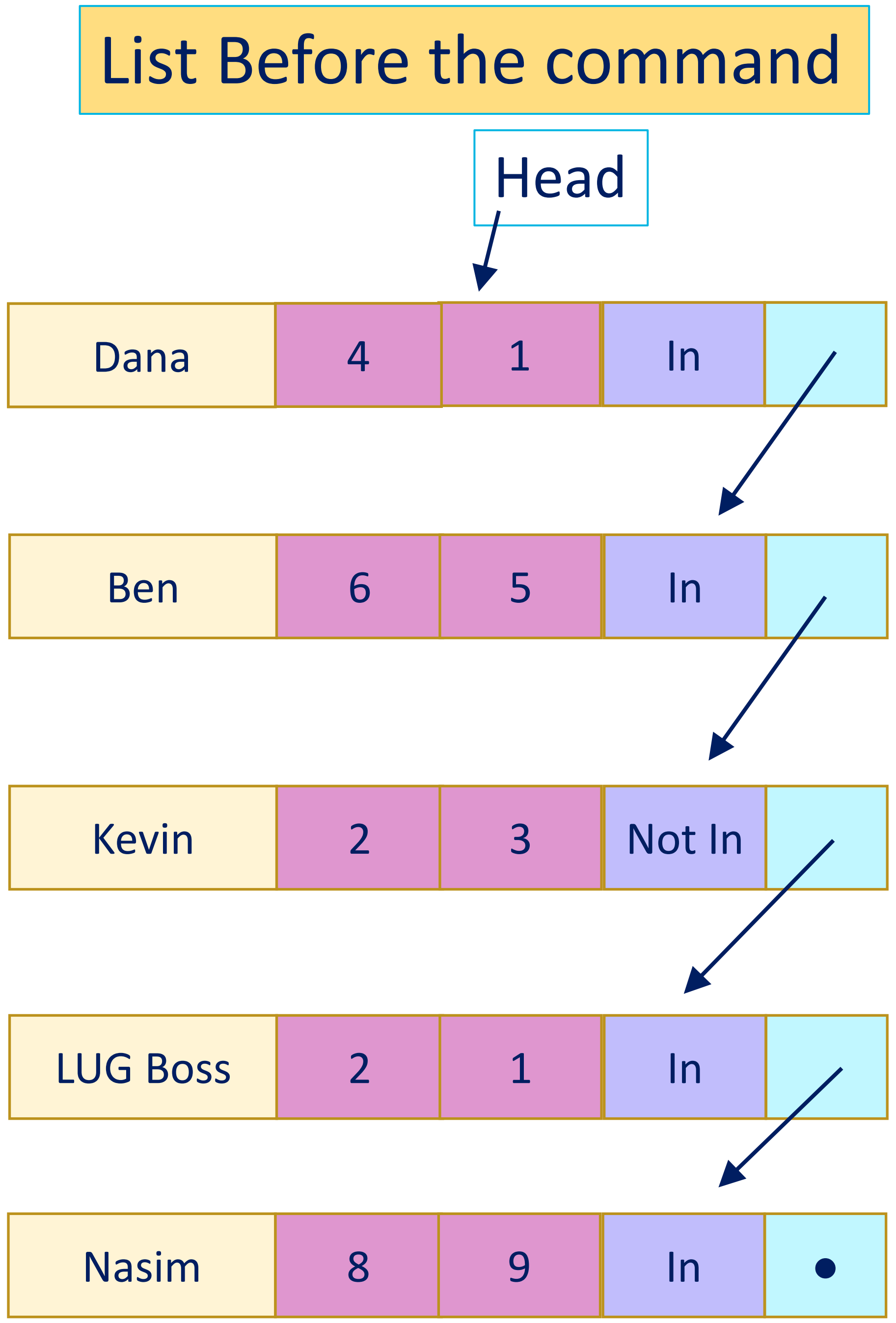
# Running the Program

- After a brief moment of panic, LUG Boss wants to know how long will it take until his order is ready so he asks for an estimated waiting time.
- Command: t LUG Boss
- The estimated wait time for LUG Boss is 78 minutes !!!!



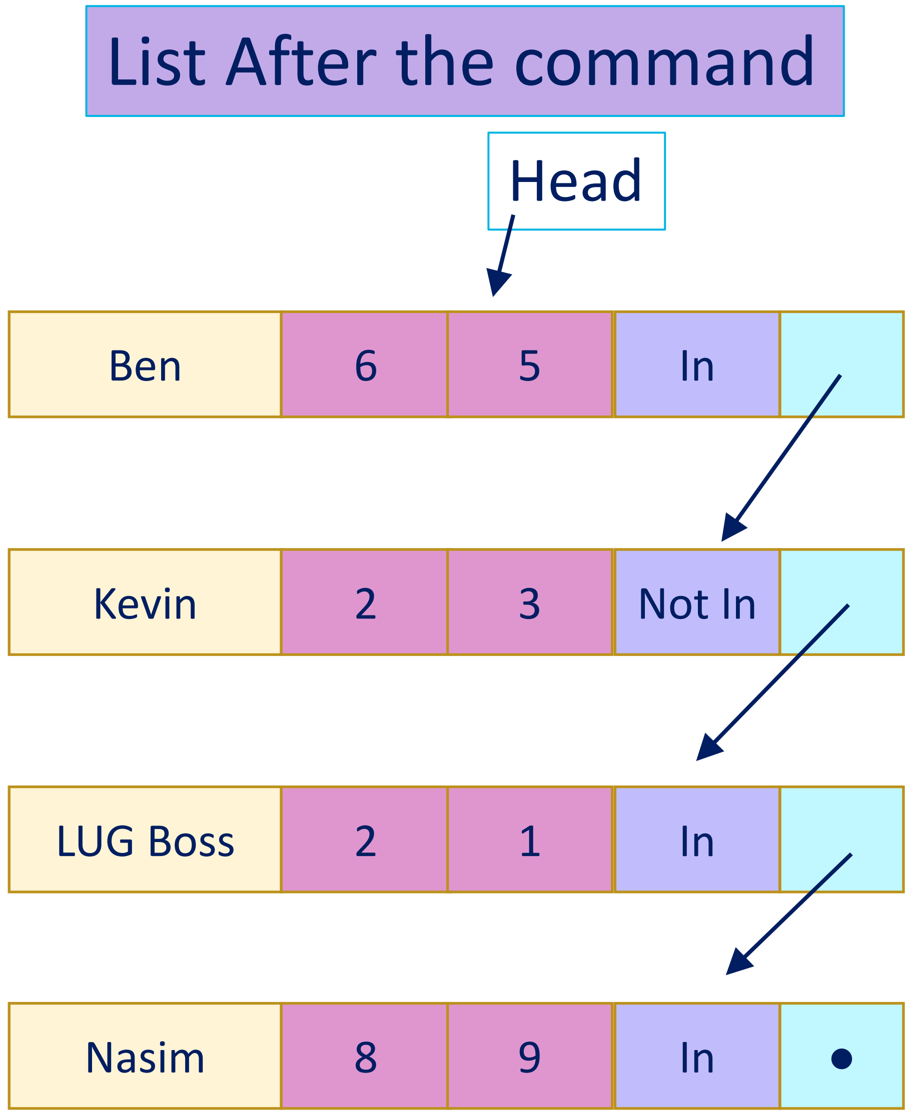
# Running the Program

- An order of 4 burgers and 2 salads is ready on the counter for pick up.
- Command: r 4 2
- Program finds first order with 4 or less burgers and 2 or less salads that is "In" the restaurant
- Dana order is selected and her name will be called for picking up her order
- Removes that node from the list



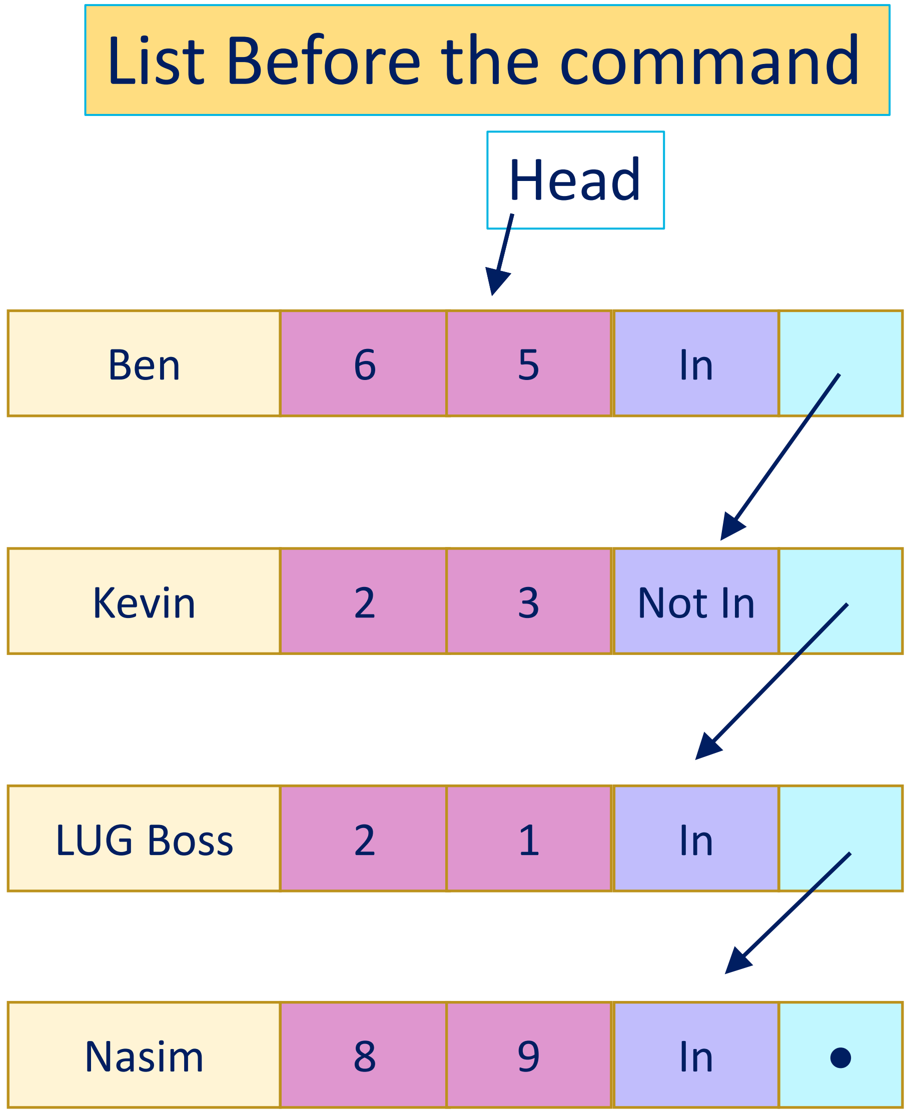
# Running the Program

- An order of 4 burgers and 2 salads is ready on the counter for pick up.
- Command: r 4 2
- Program finds first order with 4 or less burgers and 2 or less salads that is “In” the restaurant
- Dana order is selected and her name will be called for picking up her order
- Removes that node from the list



# Running the Program

- An order of 2 burgers and 4 salads is ready on the counter for pick up.
- Command: r 2 4
- Program finds first order with 2 or less burgers and 4 or less salads that is "In" the restaurant
  - Ben order has more food items
  - Kevin is NOT in the restaurant
- Calls LUG Boss to pick up his order
- Removes that node from the list

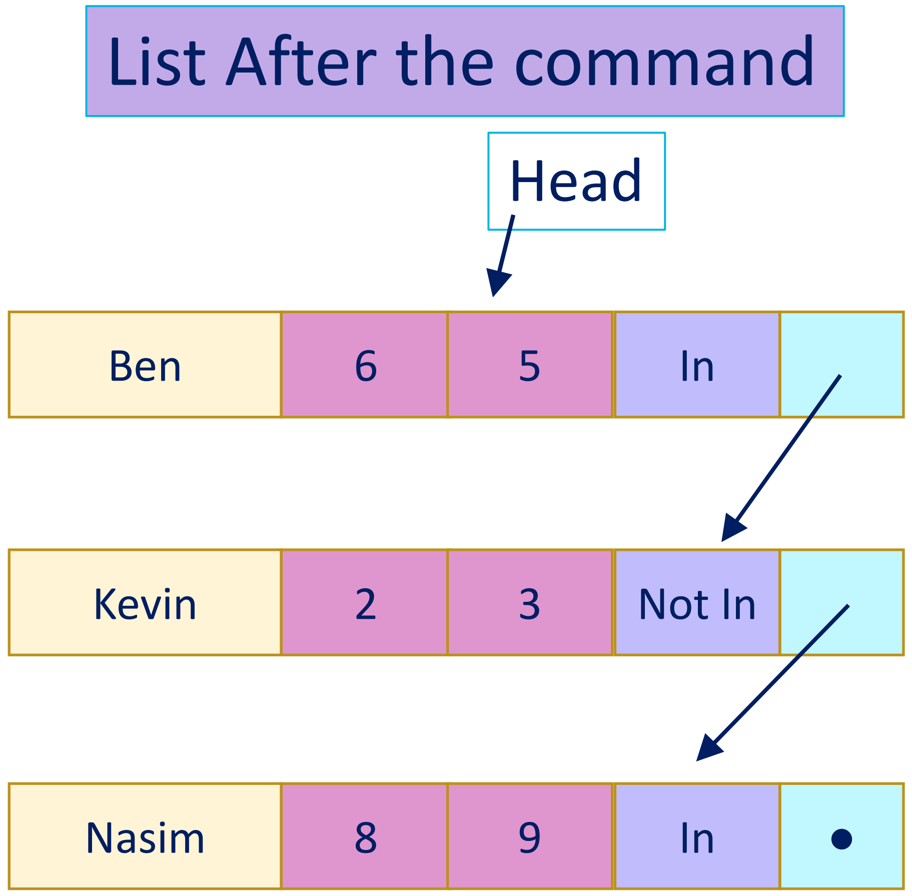


# Running the Program

- An order of 2 burgers and 4 salads is ready on the counter for pick up.
- Command: r 2 4

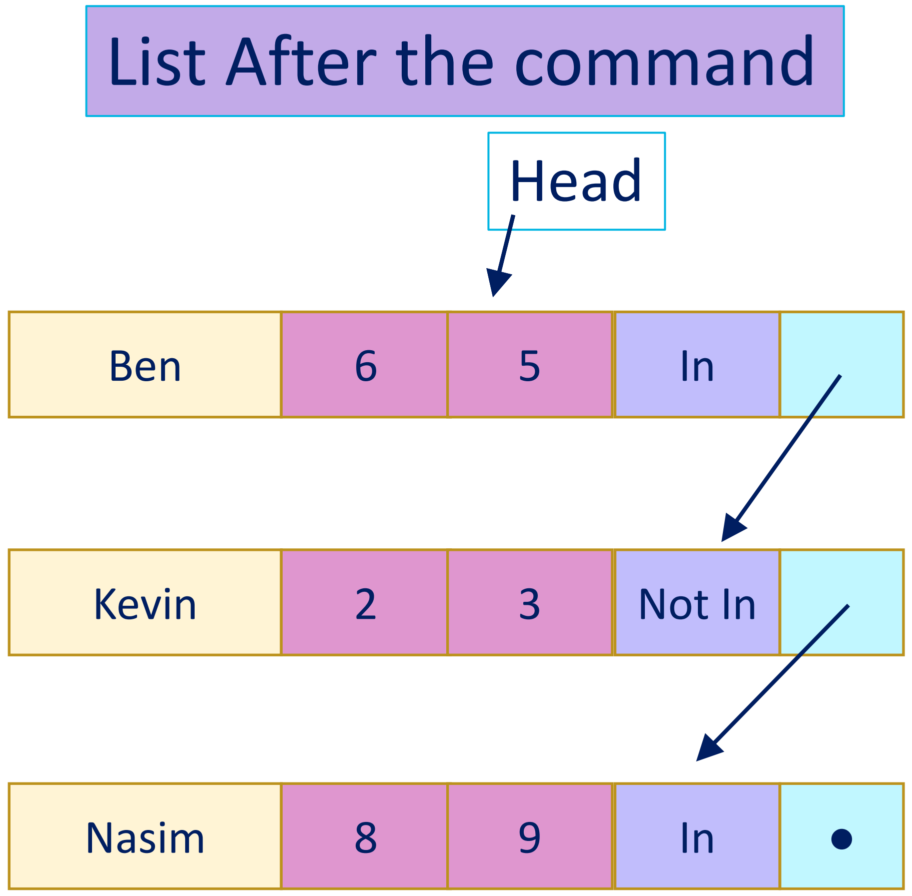
- Program finds first order with 2 or less burgers and 4 or less salads that is "In" the restaurant
    - Ben order has more food items
    - Kevin is NOT in the restaurant

- Calls LUG Boss to pick up his order
- Removes that node from the list



## Running the Program

- An order of 3 burgers and 3 salads is ready on the counter for pick up.
- Command: r 3 3
- Program finds first order with 3 or less burgers and 3 or less salads that is “In” the restaurant
  - Ben order has more food items
  - Kevin is NOT in the restaurant
  - Nasim order has more food items
- Reports no one can pick up the order
- List will not change

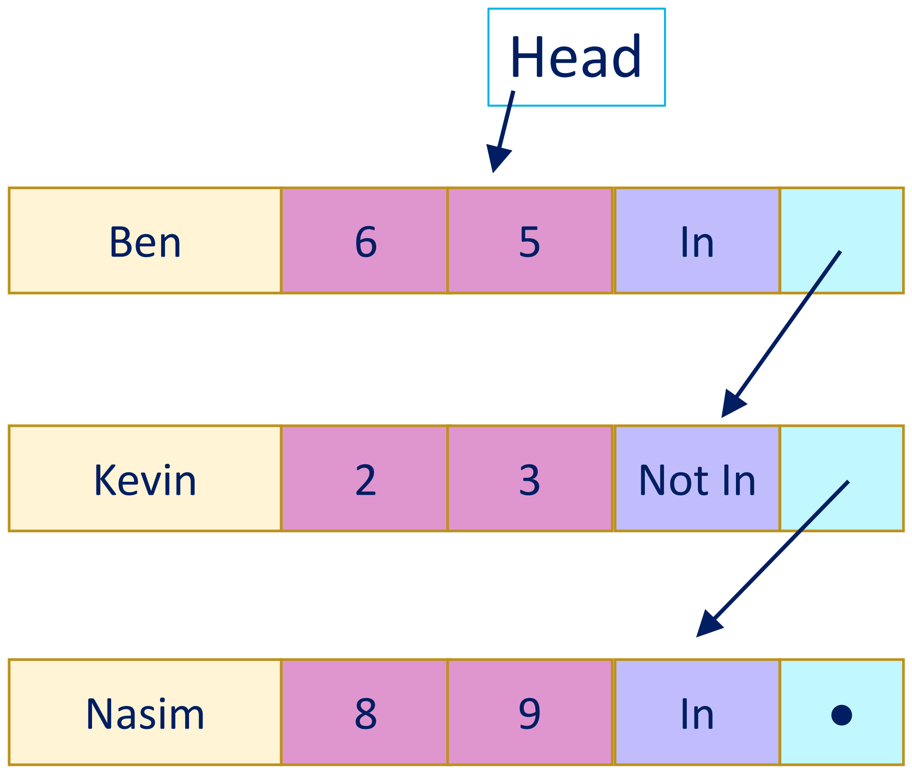


# Running the Program

- Kevin finally arrives
- Command:   w   Kevin

- Program finds the Kevin order node
- Change “Not In” to “In”

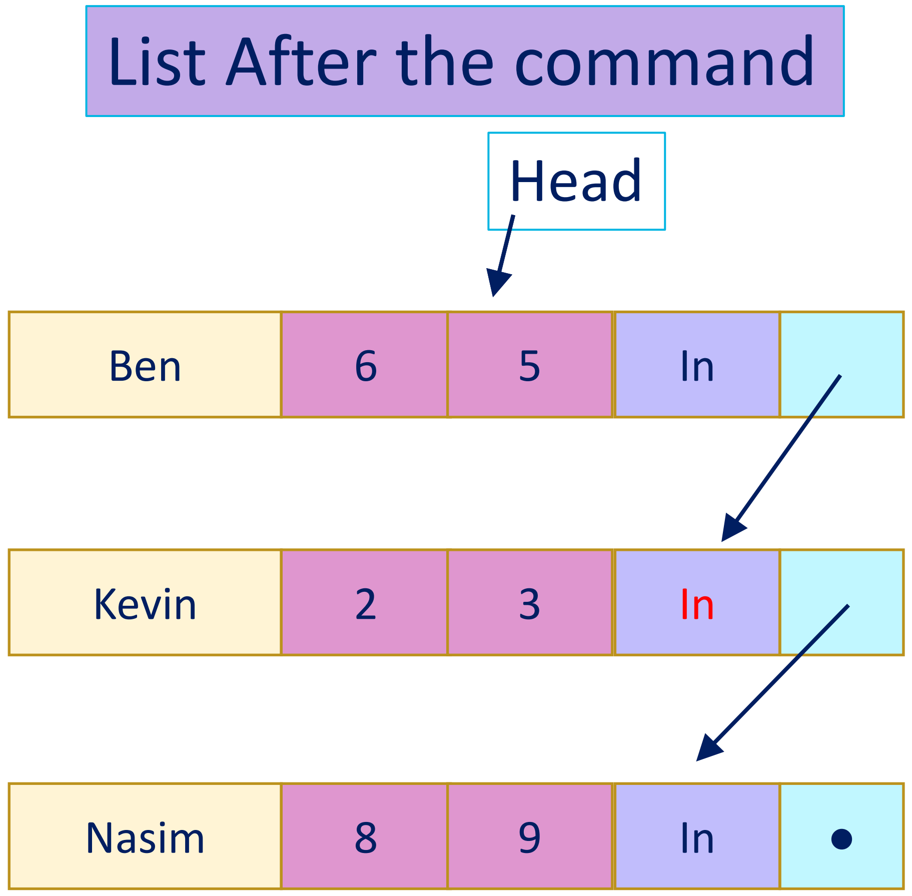
List Before the command



## Running the Program

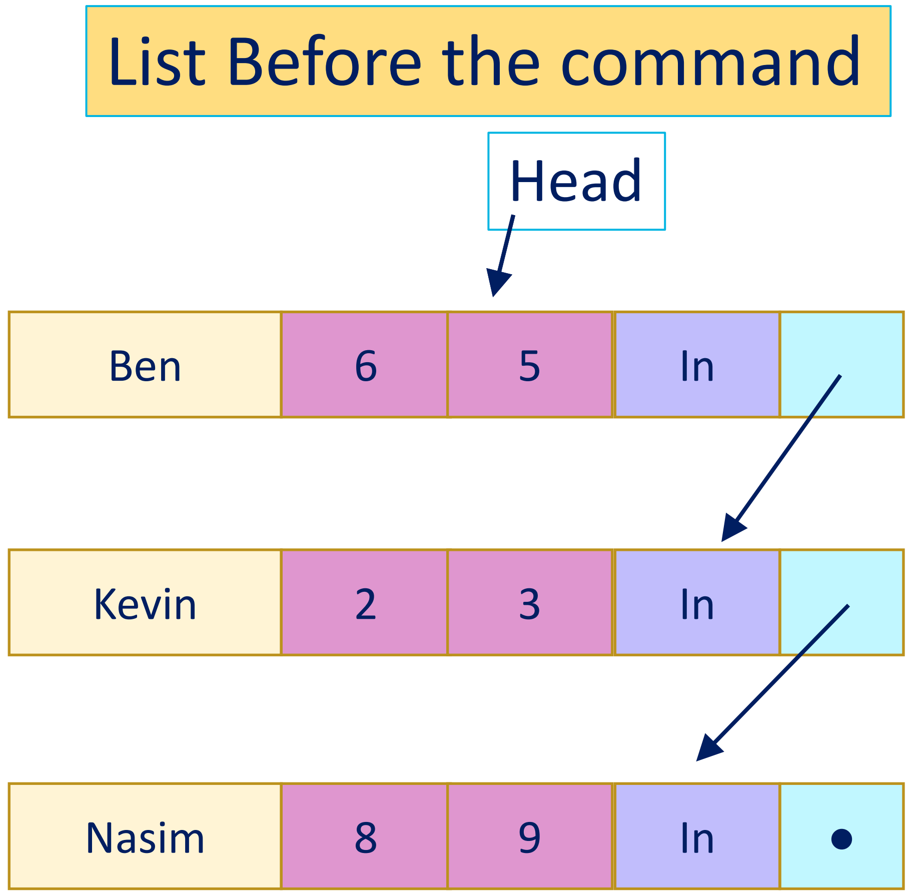
- Kevin finally arrives
- Command: w Kevin

- Program finds the Kevin order node
- Change “Not In” to “In”



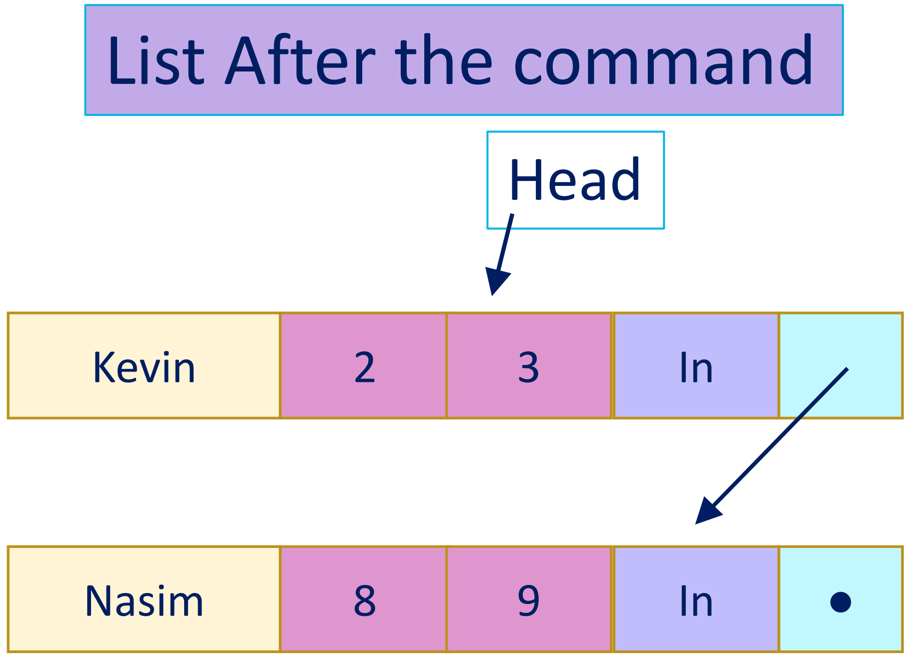
# Running the Program

- An order of 6 burgers and 5 salads is ready on the counter for pick up.
- Command: r 6 5
- Program finds first order with 6 or less burgers and 5 or less salads that is "In" the restaurant
- Calls Ben to pick up order
- Removes that node from the list



## Running the Program

- An order of 6 burgers and 5 salads is ready on the counter for pick up.
- Command: r 6 5
- Program finds first order with 6 or less burgers and 5 or less salads that is “In” the restaurant
- Calls Ben to pick up order
- Removes that node from the list

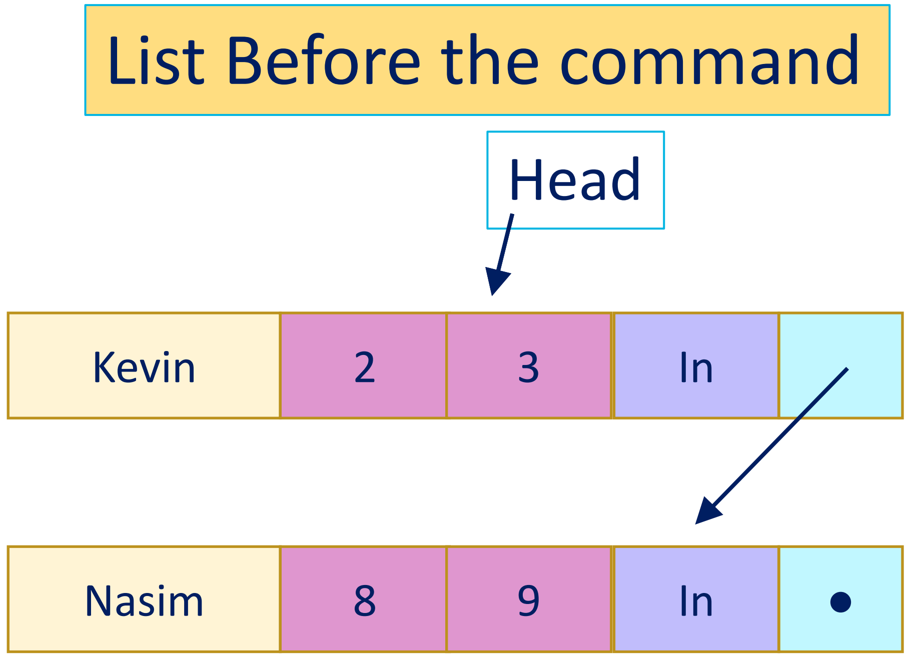


## Running the Program

- An order of 8 burgers and 10 salads is ready on the counter for pick up.
- Command:   r   8  10

- Program finds first order with 8 or less burgers and 10 or less salads that is “In” the restaurant

- Calls Kevin to pick up order
- Removes that node from the list
- Note that the program is NOT that smart

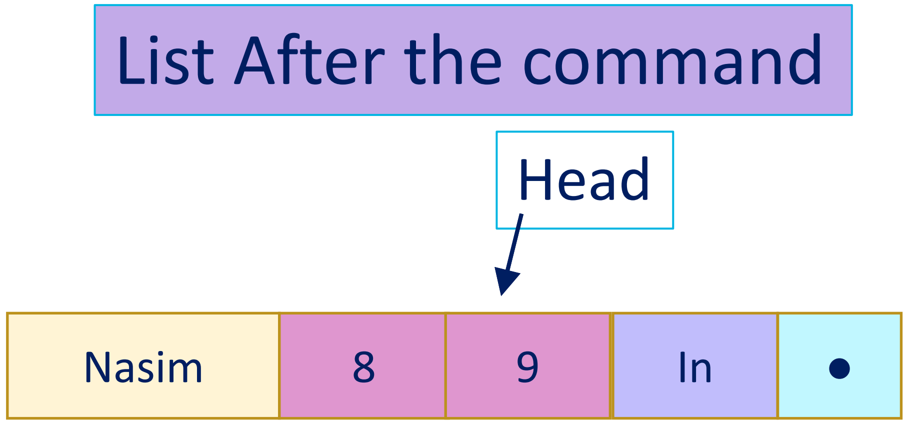


# Running the Program

- An order of 8 burgers and 10 salads is ready on the counter for pick up.
- Command: r 8 10

- Program finds first order with 8 or less burgers and 10 or less salads that is "In" the restaurant

- Calls Kevin to pick up order
- Removes that node from the list
- Note that the program is NOT that smart


List After the command
Head
Nasim
8
9
In
●