Uber Career Prep: Homework Assignment 1

Your own authentic solution is far more interesting than anything else. These questions should be completed without using the internet, with your own knowledge as it stands, and stated assumptions. You can easily Google these questions and find the optimum answers. That's not what we want. We want to gauge how you think, and see your creativity! It is infinitely more interesting to see your unique solution, assumptions, and struggles than the generic optimum answer that is on Google. So give it your best, and have fun. We encourage you to review all of the questions first, and let it brew in your mind. Don't wait till the last day to do these.

Do these steps before getting started with your homework:

1. **Review this doc. There's a Frequently Asked Questions section at the bottom of the page. It's currently empty, but with your helpful questions, we can improve it! If you have any questions, please add a comment in the FAQ section, and the assignment creators will reply!**
2. **Complete the Workshop 1 Feedback Survey**
3. **Join the Discord Group**
    - You should have received an email invite to join already
    - This space is for general communication about coding projects and homework that you are working on
    - The homework admins will be available for general questions, however, it is highly recommended that students use the Frequently Asked Questions section so others may benefit from general questions/answers:
        - **Pouria** (pouria@uber.com)
        - **James** (ormond@uber.com)
4. **Schedule a 30 minute sync with your mentor for next week, the week of 3/22 or 3/29. In this discussion you should cover the following with your mentor:**
    - Mentor will assess your skill level via conversation regarding your education & experience
    - Share the programming language(s) you are most comfortable with

- Review the homework, and set goals **here** that describe what you will achieve by the time of the next workshop (5/13) with your mentor. You and your mentor must agree on this goal before it is considered final.
- Meeting cadence: 30 minute zooms required monthly, but every 2 weeks is suggested.

Submission & Due dates

We've broken this assignment down into 4 due dates, each with a suggested milestone from the assignment to complete. Get familiar with the assignment, and then align with your mentor on your overall goals and what you plan to complete for each of the listed due dates. Note that you may be asked to make changes to your code when it is graded. Please leave time for this.

HW Submission Dates:

- **April 2nd - Parts 1 & 2**
- **April 16th - Part 3**
- **April 30th - Part 4**
- **May 12th - Part 5**

Part 1: Environment Setup (1-2 hrs)

(Pssst. You can use the internet for this part)

You should have already received an email inviting you to join the Uber Career Prep 2021 Github Organization: ubercareerprep2021. This is where you will create a repository and host your completed homework.

Learn more about github setup here if you haven't used it before: Setup and Getting Started YouTube Video (To make sure you got it, you can create a dummy folder in your local machine, and push it to your github repository. Feel free to get more familiar with github tutorials)

Once you've completed the entire homework assignment, **take these steps to submit your assignment to GitHub:**

1. Create a Github Repository named "Uber-Career-Prep-Homework-<first_name>-<last_name>"
    a. Ex: Uber-Career-Prep-Homework-James-Ormond
2. Create a sub-folder within that repository named "Assignment-1"
3. Add each file for your homework submission to the "Assignment-1" folder and name them as follows:
    a. Part1.<filetype>
    b. Part2.<filetype>
    c. Part3.<filetype>
    d. Part4.<filetype>
    e. Part5.<filetype>
4. Push your work to your repository (For tips on setting the remote for your repository, use this link)

For all questions in this homework assignment, please use the coding language you feel most comfortable with; we will provide some examples, but you are not required to use the languages in those samples.

Send an email to careerprep@uber.com with the subject "Assignment 1 | FirstName LastName"

Include a github URL that links to your assignment in the body of the email

*For the remaining milestones, you are **not allowed** to use the internet. Only your knowledge, and information provided herein. :)*

Part 2: Arrays & Strings (1 hr):

isStringPermutation(...)

Implement the function `isStringPermutation()` that takes two Strings as parameters and returns a Boolean denoting whether the first string is a permutation of the second string.

Go-Lang `func isStringPermutation(s1 string, s2 string) bool {}`

Swift `func isStringPermutation(s1: String, s2: String) -> Bool {}`

Python 3 `def isStringPermutation(s1: str, s2: str) -> bool:`

Java `public boolean isStringPermutation(String s1, String s2) {}`

Below are some examples:

1. `isStringPermutation(s1: "asdf", s2: "fsda") == true`
2. `isStringPermutation(s1: "asdf", s2: "fsa") == false`
3. `isStringPermutation(s1: "asdf", s2: "fsax") == false`

pairsThatEqualSum(...)

Implement the function `pairsThatEqualSum()` that takes an array of integers and a target integer and returns an array of pairs (i.e., an array of tuples) where each pair contains two numbers from the input array and the sum of each pair equals the target integer. (Order of the output does not matter).

Go-Lang `func pairsThatEqualSum(inputArray []int, targetSum int) []int{}`

Swift `func pairsThatEqualSum(inputArray: Array<Int>, targetSum: Int) -> Array<(Int, Int)> {}`

Python 3 `def pairsThatEqualSum(inputArray: list, targetSum: int) -> list:`

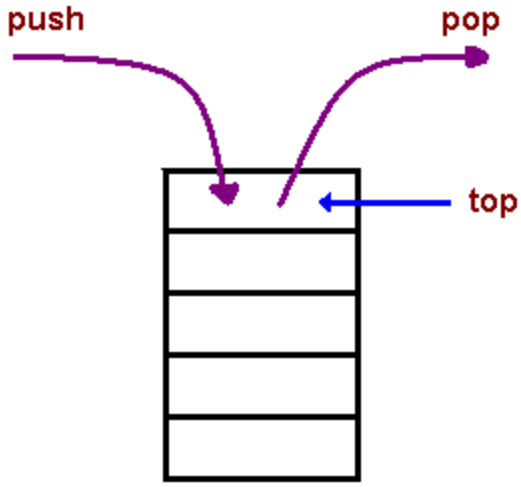
Java `public List<List<Integer>> pairsThatEqualSum(List<Integer> inputArray, Integer targetSum) {}`

Below are some examples:

1. `pairsThatEqualSum(inputArray: [1, 2, 3, 4, 5], targetSum: 5) == [(1, 4), (2, 3)]`
2. `pairsThatEqualSum(inputArray: [1, 2, 3, 4, 5], targetSum: 1) == []`
3. `pairsThatEqualSum(inputArray: [1, 2, 3, 4, 5], targetSum: 7) == [(2, 5), (3, 4)]`

Part 3: Stacks and Queues (2 hrs):

Stacks



A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In "pushdown stacks" only two operations are allowed: **push** the item onto the top of the stack, and **pop** the item off of the top of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. (Hence the name Stack, cause you stack items on top, and remove from the top.) A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.

Challenge: Implement the Stack class from scratch (do not use your language's standard stack or queue library/package methods). In this challenge, your Stack will only accept Integer values. Implement the following methods:

1. `push()` ➜ Pushes an integer on top of the stack
2. `pop()` ➜ Removes what is on the top of the stack, and returns that value to the caller
3. `top()` ➜ Looks at the top value, and returns it. Does not manipulate the stack
4. `isEmpty()` ➜ Returns True or False if the stack is Empty or not, respectively
5. `size()` ➜ Returns an integer value with the count of elements in the stack

Here is a sample execution trace:

```
myStack = Stack()

myStack.push(42)

print "Top of stack: ", myStack.top()

# prints "Top of stack: 42"

print "Size of stack: ", myStack.size()

# prints "Size of stack: 1"

popped_value = myStack.pop()
```

```
print “Popped value: ” , popped_value

# prints “Popped value: 42”

print “Size of stack: ”, myStack.size()

# prints “Size of stack: 0”
```

Make sure to write extension unit tests for your stack. Consider, for example, what would happen if you pop off more items than you push.

Bonus - if you want to push[1] yourself further, try these extra challenges!

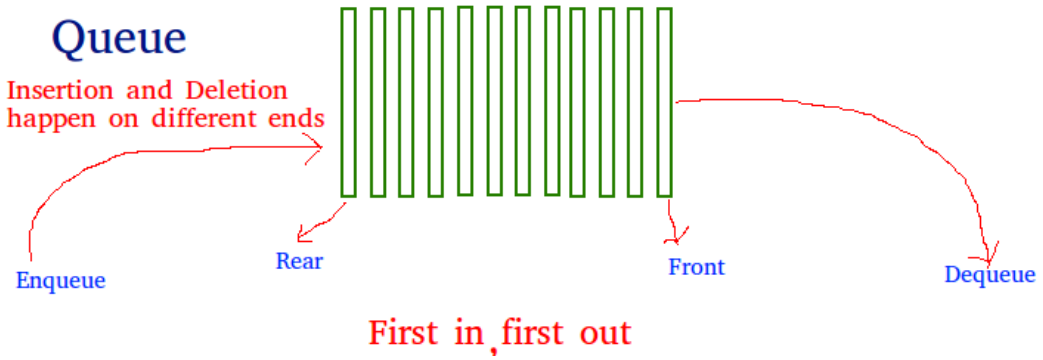
1. Add a new method to your Stack class called `min()`, which returns the minimum element of the stack in O(1) time, as opposed to O(n) time.
2. Allow your stack to handle any type of object as input type, not just integers. To further illustrate this ask, consider the following execution trace:

```
intStack = MyCustomStack<Int>()

intStack.push(42)

# notice that this stack is accepting integers

stringStack = MyCustomStack<String>()

# notice that we are using the same MyCustomStack class, except
this time we are specifying that is accepts strings

stringStack.push(“string”)
```

[1] Pun intended 😀

Queues

A **queue** is a particular kind of abstract data type in which the entities in the collection are kept in order and the principle (or only) operations on the collection are the addition of entities to the rear terminal position, known as *enqueue,* and removal of entities from the front terminal position, known as *dequeue*. This makes the queue a First-In-First-Out (FIFO) data structure.



Implement a Queue class from scratch that handles integers, with the following methods:

1. `enqueue()` ➔ adds an item to the queue
2. `dequeue()` ➔ removes an item from the queue
3. `rear()` ➔ returns the item at the end of the queue
4. `front()` ➔ returns the item at the front of the queue
5. `size()` ➔ returns the size of the queue
6. `isEmpty()` ➔ returns whether or not the queue is empty

Here is a sample execution trace:

```
myQueue = Queue()

myQueue.enqueue(1)

myQueue.enqueue(2)

myQueue.enqueue(3)

print "Size of queue: ", myQueue.size()

# prints "Size of queue: 3"
```

```
print “Front of queue: “, myQueue.front()

# prints “Front of queue: 1”

print “Rear of queue: “, myQueue.rear()

# prints “Rear of queue: 3”

dequeuedItem = myQueue.dequeue()

print “Dequeue item: “, dequeuedItem

# prints “Dequeued item: 1”
```

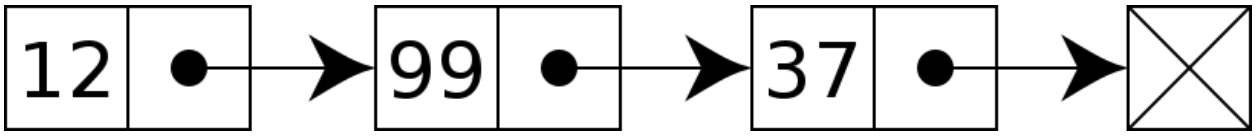
Make sure to write extension unit tests for your queue. Consider, for example, what would happen if you dequeue more items than you enqueue.

Bonus

1. Allow your queue to handle any type of object as input type, not just integers.

Part 4: Linked Lists (2 hrs)

A linked list is a linear collection of data elements, whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence. In its most basic form, each node contains: data, and a reference (in other words, a link) to the next node in the sequence. This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration. (From wikipedia)



Implement a Singly Linked List class with the following methods:

1. `void push(<Node> node)` ➔ Adds the node to the end of the list
2. `<Node> pop()` ➔ Removes the last node at the end of the linked list, returns that data
3. `void insert(uint index,<Node> node)` ➔ Adds a single node containing data to a chosen location in the list. If the index is above the size of the list, do nothing
4. `void remove(uint index)` ➔ remove/delete a single node at the index location in the list. If the node doesn't exist at the index, do nothing
5. `<Node> elementAt(uint index)` ➔ Returns a pointer to the node at the index location in the list. If the node doesn't exist at the index, return nil/null
6. `uint size()` ➔ Returns the length of the list.
7. `void printList()` ➔ Returns a string representation of the linked list

Tips:

- First define your `Node` (contains a data element of integer type, and a next-node element that is a node-pointer).
- Then define a Linked List that links the nodes as per the above methods.

Implement the following tests (should be self explanatory):

- `testPushBackAddsOneNode`

- `testPopBackRemovesCorrectNode`
- `testEraseRemovesCorrectNode`
- `testEraseDoesNothingIfNoNode`
- `testElementAtReturnNode`
- `testElementAtReturnsNoNodeIfIndexDoesNotExist`
- `testSizeReturnsCorrectSize`

In a Linked List, a “cycle” occurs if a given node in the Linked List references an earlier node for its “next” reference. Add a method to your Linked List called `hasCycle()` which returns a boolean denoting whether a cycle exists.

Bonus

Implement a function to check if a linked list is a palindrome.

Part 5: Reverse a Linked List (2 hrs)

Implement `reverseLinkedList()` which takes in a linked list and returns a new linked list with the same elements in reverse order. For example, if the representation of your Linked List is “1, 2, 3, 4”, then reversing it would return “4, 3, 2, 1”.

We have identified three main methodologies for achieving this solution, and we want to challenge you to build all three:

1. Iteratively
    a. In this context, “iteratively” means “looping through the data set”. Therefore, our solution should have a time complexity of O(n) and a space complexity of O(1).
2. Using a stack
    a. Leaning on our knowledge of stacks, can you think of a way to utilize a stack to solve this problem with a time complexity of O(n)?
3. Recursively
    a. Read up on recursion, and then apply that knowledge to come up with another version of this algorithm.

Homework Contact: Pouria Pezeshkian and James Ormond.

Discord: https://discord.com/invite/uamsJVCGcR

FAQ

If you have any questions, please leave a comment in the doc and then tag either ormond@uber.com or pouria@uber.com. Additionally, please feel free to add comments throughout the document.

1. Q: When is the assignment due?
   A: The guidelines for Homework Assignment 1 submission are bookmarked within this document: Here.

2. Q: What programming language should I use?
   A: Any! It is recommended that you stick with the programming language of choice, and not switch between different parts of the assignment. Some examples of programming languages for different use cases are (Python, Java, GoLang, Swift, ...)

3. Q: Where do I get started?
   A: Start Here, and set up your environment.

4. Q: Can I use the internet to help me find the solution?
   A: No! That would defeat the purpose of the exercises!

5. Q: For part 2, in the pairsThatEqualSum method, does order matter for the output of the pairs?
   A: Order does not matter.

6. Q: If I have a question that is not answered here, what do I do?
   A: Comment in the section right below (Google docs allows you to comment within the document, feel free to do so below). We actively monitor this document.

7. Q: If we decide to use Python for a part of the assignment, are we allowed to modify the function and parameter names to adhere to snake_case (since it looks to be all in camelCase)?
   A: As long as you stay consistent across your code!
   a. Add comment here
   b. Add comment here
   c. And here
   d. And here