HARDIK KHARE | 70765344

HW 4.1 (50 points)

A company you are working for is asking you to write a program, ***musicPlayer***, that creates a playlist of songs (and their artist) in a specific order. They want you to implement your own simple data type called ***SimplePlayList*** to act as the playlist. Each song's title and the artist should be stored together, but as separate attributes. Because space is very expensive for the company (given the high number of users they support), they want ***SimplePlayList*** to be minimalistic in terms of space (ie. it should only contain attributes that are needed). An empty playlist should use minimal memory. Your main program, ***musicPlayer***, should utilize a ***SimplePlayList*** object and be able to process the following commands by the user:

- ***push:*** adds a song to the front of the playlist in *O(1)* time.
- ***queue:*** adds a song to the end of the playlist in *O(n)* time.
- ***current:*** displays the current song, its previous song, and its next song in *O(n)* time.
- ***delete:*** deletes current song in *O(1)* time. Current moves to the next song.
- ***prev:*** makes the previous song the new current song in *O(n)* time (the "*prev*" of the first song should be the last song).
- ***next:*** makes the next song the new current song in *O(1)* time (the "*next*" of the last song should be the first song).
- ***restart:*** makes the first song in the list the new current song in *O(1)* time
- ***find***: in *O(n)* time finds the queried song, and in *O(1)* time from when the song is found, is able to display it, its previous song, and its next song.
- ***changeTo:*** changes current song to the song entered by the user in *O(n)* time
- ***addBefore:*** adds a song before another existing song in *O(n)* time
- ***addAfter:*** adds a song after another existing song in *O(n)* time
- ***random:*** makes a random song the new current song in *O(n)* time
- ***print***: displays the playlist (in order) in *O(n)* time

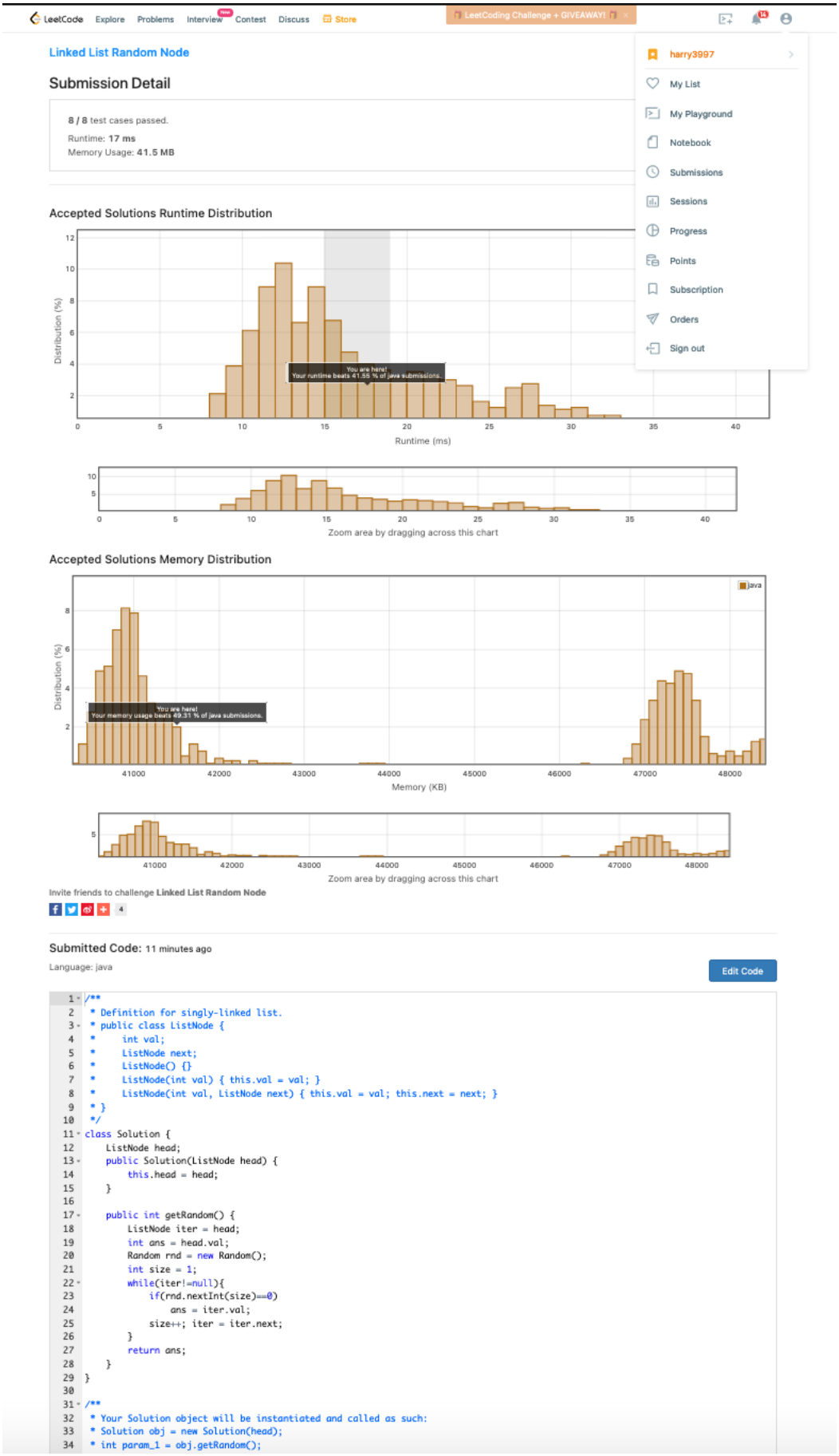
```
Please enter choice
push Mundian To Bach Ke [Panjabi MC]
Please enter choice
push My Immortal [Evanescence]
Please enter choice
queue California Love [Tupac]
Please enter choice
next
Please enter choice
current
The current song is: California Love  [Tupac]
    The previous song is: Mundian To Bach Ke  [Panjabi MC]
    The next song is: My Immortal  [Evanescence]
Please enter choice
addBefore Mundian To Bach Ke [Panjabi MC] Canta Per Me [Yuki Kajiura]
Please enter choice
addAfter Mundian To Bach Ke [Panjabi MC] Shape Of My Heart [Sting]
Please enter choice
current
The current song is: California Love  [Tupac]
    The previous song is: Shape Of My Heart  [Sting]
    The next song is: My Immortal  [Evanescence]
Please enter choice
prev
Please enter choice
prev
Please enter choice
delete
```

```
    The next song is: My Immortal  [Evanescence]
Please enter choice
prev
Please enter choice
prev
Please enter choice
delete
Please enter choice
changeTo California Love [Tupac]
Please enter choice
current
The current song is: California Love  [Tupac]
    The previous song is: Shape Of My Heart  [Sting]
    The next song is: My Immortal  [Evanescence]
Please enter choice
find Mundian To Bach Ke [Panjabi MC]
Song not found
Please enter choice
print
The playlist is:
1. My Immortal  [Evanescence]
2. Canta Per Me  [Yuki Kajiura]
3. Shape Of My Heart  [Sting]
4. California Love  [Tupac]
Please enter choice
quit

Process finished with exit code 0
```

1. What data structure did you implement ***SimplePlayList*** as?
   A circular linked list.
2. List all the attributes (aka fields) you needed in order to implement ***SimplePlayList*** (also include the attributes for any other auxiliary data structures it uses)? (Do not list functions (aka methods)).
   Attributes used: String song_title; String artist; SimplePlayList nextSong;
3. How does ***SimplePlayList*** retrieve a random song in *O(n)* time? Explain in detail using a few sentences.
   We can retrieve a random song in O(n) time using Reservoir Sampling. This approach is used to get a random value when we do not know the total number of elements.
4. If ***prev*** is processed in *O(n)* time, then how is ***find*** able to print the previous song of the found song in *O(1)* time?
   In the find function while we are looking for the desired song, we also use a 'prev' pointer to keep track of the previous song. If we find our desired song, we can use our prev pointer to print the details for previous song of found song in O(1).

https://leetcode.com/problems/linked-list-random-node/

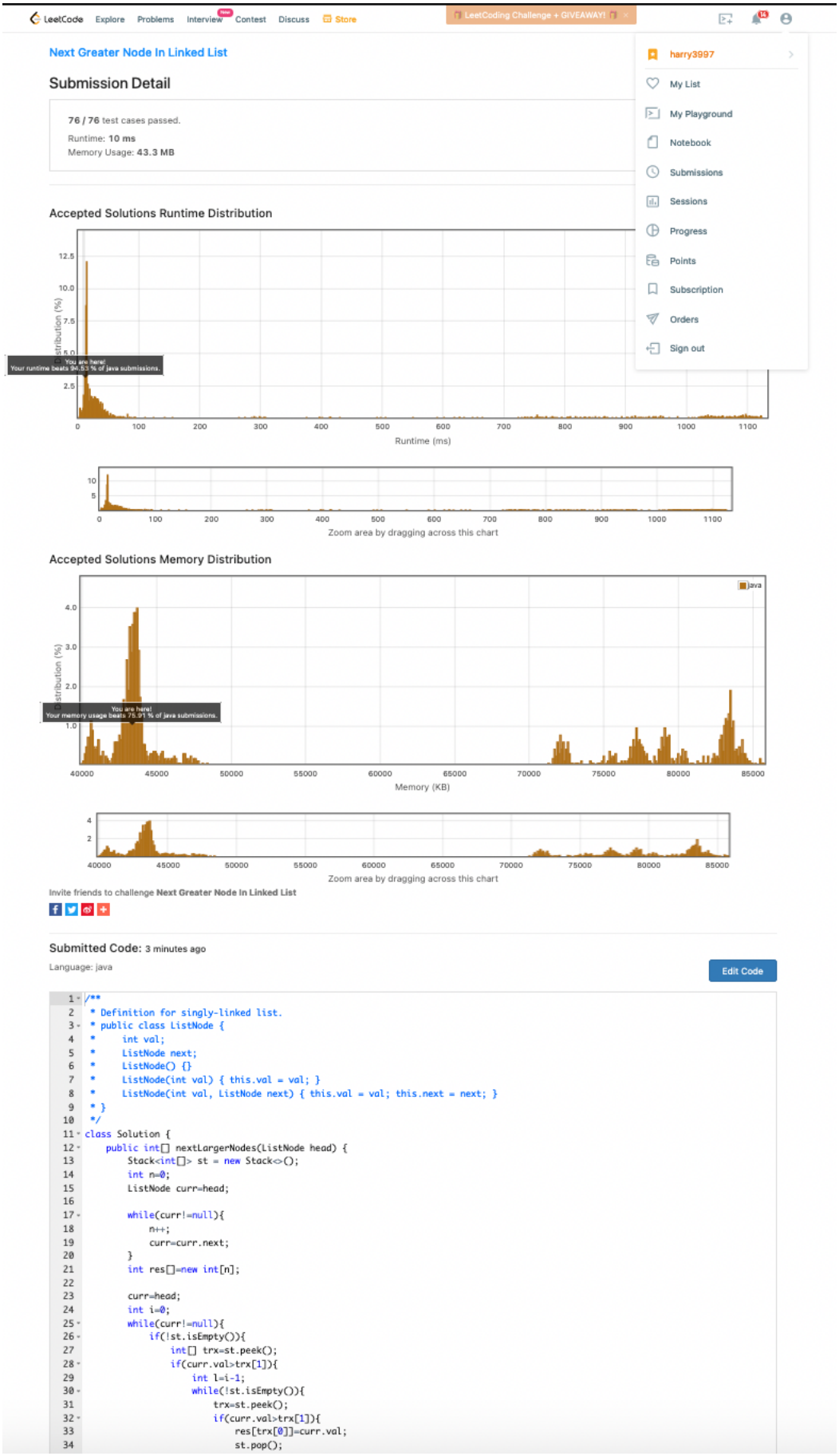



```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    ListNode head;
    public Solution(ListNode head) {
        this.head = head;
    }

    public int getRandom() {
        ListNode iter = head;
        int ans = head.val;
        Random rnd = new Random();
        int size = 1;
        while(iter!=null){
            if(rnd.nextInt(size)==0)
                ans = iter.val;
            size++; iter = iter.next;
        }
        return ans;
    }
}

/**
 * Your Solution object will be instantiated and called as such:
 * Solution obj = new Solution(head);
 * int param_1 = obj.getRandom();
```

https://leetcode.com/problems/next-greater-node-in-linked-list/




```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public int[] nextLargerNodes(ListNode head) {
        Stack<int[]> st = new Stack<>();
        int n=0;
        ListNode curr=head;

        while(curr!=null){
            n++;
            curr=curr.next;
        }
        int res[]=new int[n];

        curr=head;
        int i=0;
        while(curr!=null){
            if(!st.isEmpty()){
                int[] trx=st.peek();
                if(curr.val>trx[1]){
                    int l=i-1;
                    while(!st.isEmpty()){
                        trx=st.peek();
                        if(curr.val>trx[1]){
                            res[trx[0]]=curr.val;
                            st.pop();
```