CSC 1109 Lab 6

Woon Jun Wei, 2200624 February 9, 2023

Question 1

Listing 1: LinkList.java

```java
import java.util.*;
public class lab6_LL {

    public static void main(String[]  args){
        LinkedList<Integer> ll = new LinkedList<Integer>();

        ll.add(1);
        ll.add(3);
        ll.add(5);
        ll.add(7);
        ll.add(9);
        ll.add(11);

        // Question 1
        System.out.print("Before adding: ");
        printLinkedList(ll);
        System.out.print("After adding and sorting: ");
        printLinkedList(addAndSort(ll, 10));

        // Question 2
        //System.out.print("Before swapping:");
        //printLinkedList(ll);
        System.out.print("After swapping:");
        swap(ll, 1, 6);
        printLinkedList(ll);

        //Question 3
        LinkedList<Integer> randInt = new LinkedList<Integer>();
        Random rand = new Random();
        System.out.println("The 500 random numbers generated are: ");
        for (int i = 0; i < 500; i++) {
            randInt.add(rand.nextInt(1000, 9999));
        }

        printLinkedList(randInt);

        int randomIntegerToSearch = rand.nextInt(1000, 9999);
        System.out.println("\nAnother random number generated is " + randomIntegerToSearch);

        int index = search(randInt, randomIntegerToSearch);
        System.out.println(index);
    }

    public static void printLinkedList(LinkedList<Integer> ll){
        System.out.print("[");
        for (Integer integer : ll) {
            if (Objects.equals(integer, ll.getLast())) {
                System.out.println(integer + "]");
                break;
            }
            System.out.print(integer + ", ");
        }
    }
    public static LinkedList<Integer> addAndSort(LinkedList<Integer> ll, int value){
        //ll.add(value);
        //Collections.sort(ll); This Works

        //Alternative is to loop through the list and insert the value at the correct index
        for (int i = 0; i < ll.size(); i++) {
            if (value < ll.get(i)) {
                ll.add(i, value);
                break;
            }
        }

    return ll;
    }

    public static void swap(LinkedList<Integer> ll, int indexOne, int indexTwo){
        int temp = ll.get(indexOne);
        ll.set(indexOne, ll.get(indexTwo));
        ll.set(indexTwo, temp);
    }

    public static int search(LinkedList<Integer> list, int searchVal){
        for (Integer integer : list) {
            if (integer == searchVal) {
                System.out.println("Found " + searchVal + " at index " + ↩
                    list.indexOf(integer));
                return list.indexOf(integer);
            }
        }
        return -1;
    }
}
```

Listing 2: LinkList.java Output

```
Before adding: [1, 3, 5, 7, 9, 11]
After adding and sorting: [1, 3, 5, 7, 9, 10, 11]
After swapping:[1, 11, 5, 7, 9, 10, 3]
The 500 random numbers generated are:
[5140, 2654, 9650, 6208, 3285, 5870, 8415, 4470, 3292, 5109, 9747, 2996, ↩
    2697, 2965, 2092, 2340, 9688, 1416, 6545, 6487, 4031, 4935, 2186, ↩
    1660, 1387, 1604, 3315, 9312, 8201, 3766, 6869, 2948, 9840, 4919, ↩
    7098, 2844, 1423, 8388, 6944, 8689, 2253, 2349, 3053, 8613, 4073, ↩
    2687, 4510, 5683, 7515, 6510, 4708, 8513, 9960, 5186, 1779, 4222, ↩
    7884, 3039, 4489, 1495, 8465, 2574, 3107, 4125, 1184, 5148, 2088, ↩
    2074, 3744, 8362, 3632, 8889, 2990, 3160, 5612, 2702, 5432, 3411, ↩
    4386, 8549, 5264, 4654, 3231, 2641, 6005, 5929, 2808, 4568, 7724, ↩
    8545, 6653, 2882, 8376, 7078, 2184, 9273, 2099, 6183, 4012, 9545, ↩
    6913, 4132, 3493, 5904, 6853, 9254, 7793, 3168, 5212, 6519, 4972, ↩
    9487, 5869, 5483, 2048, 6566, 3173, 1376, 3291, 5639, 6290, 6948, ↩
    2285, 4691, 8525, 6715, 4300, 7707, 1212, 7013, 2793, 2816, 5154, ↩
    6177, 2513, 5160, 1169, 3522, 9254, 8663, 5847, 2297, 6056, 3298, ↩
    4464, 1348, 3088, 4748, 2952, 9422, 2408, 1016, 2278, 2899, 9530, ↩
    2531, 2884, 5474, 8693, 1192, 6024, 3681, 8855, 6679, 1082, 1591, ↩
    5889, 1675, 3996, 6724, 1923, 9263, 2845, 6660, 8871, 4396, 1866, ↩
    2981, 3851, 4887, 6493, 1801, 3104, 1817, 1974, 6150, 5140, 3885, ↩
    9704, 7804, 6548, 8148, 8148, 1882, 6748, 5225, 4651, 8040, 7697, ↩
    6017, 2045, 2335, 1186, 5728, 8235, 2649, 2974, 2281, 5102, 1182, ↩
    1483, 8741, 6740, 8155, 3097, 8990, 8104, 2531, 2465, 3053, 3527, ↩
    1877, 7297, 8537, 7119, 7861, 7527, 6972, 3059, 2964, 9109, 8366, ↩
    5064, 8960, 9113, 4413, 3397, 9594, 2844, 2537, 8949, 8363, 1335, ↩
    7825, 5092, 1487, 4303, 3024, 7311, 3089, 8365, 3269, 9683, 3195, ↩
    1920, 8125, 4941, 3182, 1478, 3290, 8295, 5685, 7451, 5847, 5367, ↩
    3331, 7911, 2760, 8549, 7343, 9024, 6815, 6924, 2094, 2998, 3157, ↩
    4569, 2093, 4675, 6744, 2895, 8084, 8614, 8453, 9492, 8584, 9370, ↩
    6607, 2248, 9621, 3197, 7886, 5800, 7759, 9254, 2246, 6267, 7150, ↩
    5415, 6560, 9630, 7852, 8695, 2926, 7301, 2082, 4559, 7903, 3526, ↩
    2740, 1180, 6096, 5915, 3697, 1062, 2550, 9672, 8396, 7711, 1517, ↩
    9738, 5090, 6433, 2905, 4780, 4820, 3003, 3276, 1072, 2252, 1503, ↩
    5251, 6978, 4992, 4922, 1889, 4002, 3729, 5753, 4632, 3582, 6036, ↩
```

```
   2606, 8320, 3832, 1757, 8862, 9066, 1717, 7205, 4933, 1431, 9813, ↩
   2974, 4707, 3334, 8474, 4348, 4306, 1800, 8563, 3164, 1419, 9400, ↩
   8810, 7017, 3922, 2391, 7801, 2272, 6614, 9316, 6509, 1759, 2940, ↩
   6521, 2784, 2886, 8195, 7072, 4500, 2884, 6492, 1676, 7204, 6054, ↩
   9121, 4462, 6390, 5965, 4183, 8756, 7622, 8132, 8743, 3366, 1251, ↩
   1229, 6151, 3927, 5405, 7189, 6319, 1256, 4355, 9667, 2299, 4497, ↩
   8437, 5226, 5316, 5378, 9691, 1596, 9387, 7033, 5787, 1722, 6389, ↩
   1815, 3883, 2641, 3488, 8028, 8874, 4700, 7825, 1715, 4137, 5165, ↩
   6384, 6476, 7985, 2361, 1378, 9432, 7383, 4831, 1267, 5819, 2865, ↩
   9034, 4533, 2343, 9981, 7449, 9535, 7119, 4554, 7902, 3630, 9828, ↩
   5574, 3708, 7472, 5022, 6656, 3721, 9429, 3682, 3080, 7998, 1930, ↩
   1860, 3590, 1056, 8884, 8227, 6516, 3601, 8520, 1454, 2728, 6416, ↩
   5097, 9983, 7685, 2223, 9105, 2145, 1604, 7275, 7339, 6000, 7857, ↩
   9790, 8742, 6900, 4998, 2831, 1877, 1675, 2051, 6158, 8537, 3502, ↩
   8439, 4232, 2970, 9007]

Another random number generated is 9175
-1
```

Question 2

Listing 3: HashMap.java

```java
import java.util.*;

public class lab6_Hash {

    public static void main(String[]  args){
        HashMap<Integer, Integer> ht = new HashMap<>();

        ht.put(0,1);
        ht.put(1,3);
        ht.put(2,5);
        ht.put(3,7);
        ht.put(4,9);
        ht.put(5,11);



        // Question 1
        System.out.print("Before adding: ");
        printHashMap(ht);
        System.out.print("After adding and sorting: ");
        printHashMap(addAndSort(ht, 10));

        // Question 2
        System.out.println("After swapping");
        swap(ht, 1, 6);
        printHashMap(ht);

        //Question 3
        HashMap<Integer, Integer> ranIntHash = new HashMap<>();
        Random rand = new Random();
        for (int i = 0; i < 500; i++) {
            ranIntHash.put(i, rand.nextInt(1000, 9999));
        }
        System.out.println("The 500 random numbers generated are: ");
        printHashMap(ranIntHash);

        int randomIntegerToSearch = rand.nextInt(1000, 9999);
        System.out.println("\nAnother random number generated is " + ↩
            randomIntegerToSearch);

        int index = search(ranIntHash, randomIntegerToSearch);
        System.out.println(index);
    }

    public static void printHashMap(HashMap<Integer, Integer> ht){
        System.out.print("[");
        for (int i = 0; i < ht.size(); i++) {
            if (Objects.equals(ht.get(i), ht.get(ht.size() - 1))) {
                System.out.println(ht.get(i) + "]");
                break;
            }
            System.out.print(ht.get(i) + ", ");
        }
    }

    public static HashMap<Integer, Integer> addAndSort(HashMap<Integer, Integer> ht, int value){
        //Collections.sort(ht); Does Not Work
        // For each value in ht, if value is less than value, insert value at that index
        for (int i = 0; i < ht.size(); i++) {
            if (value < ht.get(i)) {
                ht.put(ht.size(), ht.get(ht.size() - 1));
                // Shift all values to the right of i to the right
                for (int j = ht.size() - 2; j > i; j--) {
                    ht.put(j, ht.get(j - 1));
                }
                ht.put(i, value);
                break;
            }

        }

        return ht;
    }

    public static void swap(HashMap <Integer, Integer> ht, int indexOne, int indexTwo){
        // Swap the values at indexOne and indexTwo
        int temp = ht.get(indexOne);
        ht.put(indexOne, ht.get(indexTwo));
        ht.put(indexTwo, temp);
    }

    public static int search(HashMap <Integer, Integer> ht, int searchVal){
        // Return index of searchVal if found, else return -1
        for (int i = 0; i < ht.size(); i++) {
            if (ht.get(i) == searchVal) {
                return i;
            }
        }
        return -1;
    }
}
```

Listing 4: HashMap.java Output

```
Before adding: [1, 3, 5, 7, 9, 11]
After adding and sorting: [1, 3, 5, 7, 9, 10, 11]
After swapping
[1, 11, 5, 7, 9, 10, 3]
The 500 random numbers generated are:
[7606, 4198, 4276, 8580, 7185, 1550, 6018, 9282, 1320, 8666, 3682, 8337, ↩
    9745, 5519, 7117, 5320, 2911, 7779, 8901, 3576, 3489, 3918, 5158, ↩
    5767, 1395, 4137, 9054, 3402, 5522, 4599, 6531, 1672, 7938, 5141, ↩
    5111, 8044, 2589, 8616, 7610, 1656, 7196, 9167, 8137, 7973, 6642, ↩
    9944, 2251, 3728, 6113, 6484, 2363, 1541, 8455, 5699, 6115, 9768, ↩
    5560, 8505, 3596, 2276, 4169, 2980, 5803, 2475, 6017, 3230, 4428, ↩
    3002, 3439, 2969, 7542, 8097, 5038, 1168, 6683, 4387, 4053, 4353, ↩
    2561, 7144, 6977, 2815, 7440, 4012, 8442, 7500, 8250, 9178, 6535, ↩
    5550, 8498, 1509, 7671, 1196, 7372, 2979, 8323, 6679, 7594, 2858, ↩
    1493, 7237, 1190, 9001, 9653, 1197, 2399, 7212, 5594, 9258, 9912, ↩
    7660, 3775, 6058, 4870, 8644, 8009, 3493, 2154, 5853, 9111, 5719, ↩
    2819, 2427, 6430, 3178, 4578, 8065, 3181, 9754, 2461, 1664, 1065, ↩
    6709, 5965, 7019, 5408, 4910, 3581, 9569, 3041, 6541, 3727, 2855, ↩
    5925, 4933, 5857, 9384, 1579, 8652, 4429, 8008, 7679, 7766, 5219, ↩
    3135, 9809, 8174, 1512, 7082, 9372, 2275, 1510, 6569, 9900, 1956, ↩
    8527, 1889, 1062, 4517, 8568, 5561, 5088, 8729, 9082, 4388, 8821, ↩
    3514, 1350, 2022, 6163, 6149, 8740, 5193, 4359, 6804, 7682, 1883, ↩
    5114, 8859, 5888, 8917, 8888, 8838, 7074, 5897, 1425, 1417, 5360, ↩
    3652, 8624, 3339, 1497, 6503, 8791, 3499, 5362, 1415, 1954, 9325, ↩
    7909, 5220, 8422, 5227, 2398, 9059, 9868, 6811, 1148, 3049, 3202, ↩
    1367, 5193, 9612, 1716, 3476, 3751, 9261, 1791, 3598, 9949, 7567, ↩
    3234, 5227, 6091, 9223, 6663, 9527, 3747, 6962, 2642, 7850, 4530, ↩
    5711, 7435, 6896, 9081, 8683, 3124, 6691, 2625, 9444, 5781, 8965, ↩
    1953, 3084, 5215, 5745, 5448, 6710, 6482, 7937, 5487, 9130, 6340, ↩
    4144, 4257, 4228, 6299, 5861, 5984, 1286, 9481, 4909, 8908, 1689, ↩
    2240, 8113, 5134, 8428, 1296, 9206, 2382, 7786, 8356, 5399, 8712, ↩
    3676, 1942, 5100, 5346, 2929, 4966, 4612, 8587, 6916, 7630, 5597, ↩
    6134, 4507, 8512, 4782, 8147, 8763, 3698, 7896, 8663, 7661, 9060, ↩
    9757, 9811, 4351, 4171, 9060, 6882, 9099, 5857, 1857, 3567, 5534, ↩
    1795, 7141, 4051, 1324, 5898, 5387, 4393, 8682, 5117, 6216, 5507, ↩
    7927, 3160, 9319, 8227, 7456, 9495, 1169, 8867, 1494, 1053, 8256, ↩
    9698, 5844, 7399, 6841, 1805, 7273, 6921, 7885, 1397, 2238, 1589, ↩
    8858, 8821, 2298, 5419, 4156, 9401, 9434, 6104, 1335, 9990, 2109, ↩
    4885, 4129, 1290, 3527, 9021, 8575, 4381, 1771, 2425, 9850, 8656, ↩
    7421, 9177, 2525, 4816, 5808, 2838, 8942, 3375, 8391, 5823, 6589, ↩
    1270, 4564, 3962, 1661, 6562, 5324, 4463, 7308, 4539, 8346, 3699, ↩
    2016, 4619, 9447, 9338, 7002, 3790, 1197, 3944, 4517, 3192, 8291, ↩
    7833, 9725, 4981, 8191, 5600, 6463, 6453, 4346, 1438, 4136, 9413, ↩
    1358, 5582, 7457, 8920, 9506, 8360, 5092, 1332, 3405, 1783, 1681, ↩
    9989, 4926, 8995, 6766, 5558, 2913, 4523, 5819, 7962, 5223, 9549, ↩
    5783, 5665, 5713, 2897, 2142, 7284, 5442, 7908, 6458, 7641, 3259, ↩
    5826, 6234, 4535, 3556, 1435, 5715, 3683, 4237, 2381, 6668, 7494, ↩
    1340, 5201, 8728, 8671, 4965, 1067, 7546, 1978, 5781, 8997, 8441, ↩
    1199, 3915, 4484, 8278, 7813, 9373, 1209, 7920, 5873, 5338, 7532, ↩
    8476, 9402, 6108, 1139, 1439, 4465, 2731, 8226, 1429, 7648, 1022, ↩
    6853, 6570, 7177, 4989]

Another random number generated is 1101
-1
```