JAVA

1. Java destructor? No how to force it? System.gc();
2. How Java garbage collection works, is it guaranteed to work
3. Mutex
4. "==" vs equals():

String a = "foo";
String b = "foo";
System.out.println(a == b);
System.out.println(a.equals(b));

5. what happens when compiling and running Java code?
6. hashtable 和hashmap
7. polymorphism，子类覆盖父类的情况，父类没有子类方法的情况
8. 泛型
9. What is finalize()
10. Linked list vs Dynamic array() - for deleting element in middle
11. you have some classes - use them another project .jar files and import.
12. *Why Java Generic

A type or method to operate on objects of various type while providing compiling-time type safety that allows programmers to catch invalid types at compile time.

13. *Difference between stack memory and heap memory

Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer's RAM .

14. 线程类的实现方法(runnable interface 和 extend thread class) MyThread implements Runnable

15. 在Linux平台上运行的java code在win上能不能运行(可以 JVM)
16. 有一个题是关于interface里面method的access modifier

Python

1. Tuple VS List
2. a = [[]]*4, a[0].append(0), output
3. a = "mathworks"

print a[-5:-1:-2]

4. is there any access specifier in python? NO!
5. string 能否直接改变： NO!

s = "abc"
s[0] = "b";
print s //error!
问怎么改是正确的： s = "b"+s[1:];

6. python多继承： c同时继承a, b, 如何实现：class c(a, b), 如果a和b都有同一个函数foo(), 发生神马
7. python浅拷贝：

```
a = [1,2,'q',3]
b = a
a.append(5)
print a == b
```

8. diff between (what is)module and package;
9. 有一个题是这样的：一个class A,

```
def __init__(self,id):
  self.id=id
  id=666
```

之后a=A(123), 问a.id是啥。123
10. 解释python里的method overloading NO!
11. Python可以有multiple constructor吗 NO！
12. 如何才能使用别的package中定义的变量 **From package import Var**

concepts:

1. O(n^2), (a) O(nlogn), (b) O(n^3), (c) O(n log^3n) a,b,c与给的O(n^2)的关系

2. BST和BT的搜索时间复杂度，以及给出一个O(nlogn)的搜索算法

3. 还问了我一个从来没在面经里看到过的问题： NP complete概念
4. 问你什么是polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

5. 什么是OOP，概念是什么

OOP, is an approach to problem-solving where all computations are carried out using objects.

6. Thread - Shared memory vs Process - Non-shared memory- App
7. 什么叫Graph? Formally, a *graph* is an object consisting of a vertex set and an edge set. What kind of data structure can be used to present a Graph? Adjacency List or Adjacency Matrix

Math:

1. 班里40个学生分别上英语课和德语课（有且只有而且至少上一门）。22个上德语课，12个上英语和德语课，问多少人上英语课。
2. 32bits的singed的整数最大能到多少？2^31-1
3. 谈谈矩阵invertible,他希望你回答determinant=0
4. 给你一个3*3 矩阵，让你判断是不是singular，为什么
5. probability的题目，因为那个印度小哥对概率挺了解，所以自己给了道题，比较长的应用题，核心是conditional probability的公式
6. connected graph最少几条edge
7. 什么是四色问题

OOD

1. OOD看了一圈有ATM, restaurant reservation system, parking lot system，还有一个大哥提到了设计一个google doc
2. Design Restaurant -

Classes - menu item - food, drink, name
menu - menu items
Table, - used tables = , free tables = ,
Food, bevarage - price, supply used
Supply = Order Supply()
Receipt =
Customer =

3. ATM

Atm
Transaction
Card
cashIn
cashOut
Screen/keyboard
Database

```
public abstract class Vehicle {
4 protected ArrayList<ParkingSpot> parkingSpots new ArrayList<ParkingSpot>();
5 protected String licensePlate;
6 protected int spotsNeeded;
7 protected VehicleSize size;
8
9 public int getSpotsNeeded() { return spotsNeeded; }
10 public VehicleSize getSize() { return size; }
11
12 /* Park vehicle in this spot (among others, potentially) */ 13 public void parkinSpot(ParkingSpot s) {
parkingSpots.add(s); }
/* Remove car from spot, and notify spot that it's gone */
public void clearSpots() { ... }
21 }
public class Bus extends Vehicle
public class Car extends Vehicle
public class Motorcycle extends Vehicle
public class ParkingLot {
2 private Level[] levels;
3 private final int NUM_LEVELS 5;
4
5 public ParkingLot() { ... }
6
7 /* Park the vehicle in a spot (or multiple spots). Return false if failed. */
8 public boolean parkVehicle(Vehicle vehicle) { ... }
9 }
```

Coding

1. LC617

```
public class Solution {
    public TreeNode mergeTrees(TreeNode t1, TreeNode t2) {
        if(t1==null&&t2==null){return null;}
        int val = (t1==null?0:t1.val)+(t2==null?0:t2.val);
        TreeNode node = new TreeNode(val);
        node.left = mergeTrees((t1==null?null:t1.left),(t2==null?null:t2.left));
        node.right = mergeTrees((t1==null?null:t1.right),(t2==null?null:t2.right));
        return node;
    }
}
```

2. 给定一个函数f(a,n)，可以把一个array a的前n位反过来。比如f([1,3,4,2], 2)的话就会把a变成[3,1,4,2]。问怎么用这个函数来对数组排序。我当时是从前往后排序，每一步要三次f的操作。后来面完发现可以从后往前排序，这样每次只要2次f。比如说[1,3,4,2]，你先找最大的，是4，用一次f(a,3)把4换到最前面，再用一次f(a,4)把4换到最后。之后就不用考虑最后一位，对前面的类似做。比如假设你第一次做完是[3,2,1,4]，那你第二次就f(a,1),f(a,3)就可以把第二大的3换到倒数第二位。

```
for(int i = a.length-1;i>=0;i--){
        int max =0;
        for(int j=i;j>=0;j--){
                if(a[max]<a[j]){
                max=j;
                }
        }
        f(a,max);
        f(a,i);
        }
```

3. 用俩stack设计queue

```
public MyQueue() {
        Stack<Integer> input = new Stack();
        Stack<Integer> output = new Stack();
    }

    /** Push element x to the back of queue. */
    public void push(int x) {
        input.push(x);
    }

    /** Removes the element from in front of queue and returns that element. */
```

```
    public int pop() {
        int i = peek();
        output.pop();
        return i;
    }

    /** Get the front element. */
    public int peek() {
        if (output.empty())
            while (!input.empty())
                    output.push(input.pop());
        return output.peek();
    }

    /** Returns whether the queue is empty. */
    public boolean empty() {
        return input.empty() && output.empty();
   }
```

4. LC503 Next Greater Element II

```
public int[] nextGreaterElements(int[] nums) {
        int[] res = new int[nums.length];
        Stack<Integer> stack = new Stack<>();
        for (int i = 2 * nums.length - 1; i >= 0; --i) {
        while (!stack.empty() && nums[stack.peek()] <= nums[i % nums.length]) {
                stack.pop();
            }
        res[i % nums.length] = stack.empty() ? -1 : nums[stack.peek()];
        stack.push(i % nums.length);
        }
        return res;
   }
```

5. reverse linked list

```
        public ListNode reverseList(ListNode head) {
      return reverse(null,head);
        }

    private ListNode reverse(ListNode l, ListNode r){
        if(r==null){return l;}
        ListNode next = r.next;
        r.next = l;
        return reverse(r,next);
    }
```

6. 给一列数，找到比每个数大的最近的数, closest larger element
7. Prime N -> Example: 3231 % 2 == 0 not prime. for (int i = 3; i <= (int) sqrt(n); i += 2)

```
void sieveOfEratosthenes(int n)
    {

        boolean prime[] = new boolean[n+1];
        for(int i=0;i<n;i++)
            prime[i] = true;

        for(int p = 2; p*p <=n; p++)
        {
            if(prime[p] == true)
            {
                for(int i = p*2; i <= n; i += p)
                    prime[i] = false;
            }
        }
        for(int i = 2; i <= n; i++)
        {
            if(prime[i] == true)
                System.out.print(i + " ");
        }
    }
```

8. 用python写任意一个sort, Sort的方法和算法复杂度

```
def sort(array=[12,4,5,6,7,3,1,15]):
        less = []
        equal = []
        greater = []

        if len(array) > 1:
        pivot = array[0]
        for x in array:
        if x < pivot:
        less.append(x)
        if x == pivot:
        equal.append(x)
        if x > pivot:
        greater.append(x)
        return sort(less)+equal+sort(greater)
        else:
        return array
```

9. GCD

```
private int GCD(int a, int b) {
        if(b==0) return a;
return a % b == 0 ? b : GCD(b, a % b);
}
```

10. 给定一个数组，如何将里面出现次数大于1的元素删掉。楼主用的是hashmap

11. Stack by two Queue: 225

```
private LinkedList<Integer> q1 = new LinkedList<>();
        /** Initialize your data structure here. */
        public MyStack() {

        }

        /** Push element x onto stack. */
        public void push(int x) {
        q1.add(x);
        int sz = q1.size();
        while (sz > 1) {
        q1.add(q1.remove());
        sz--;
        }
        }

        /** Removes the element on top of the stack and returns that element. */
        public int pop() {
        return q1.remove();
        }

        /** Get the top element. */
        public int top() {
        return q1.peek();
        }

        /** Returns whether the stack is empty. */
        public boolean empty() {
        return q1.isEmpty();
        }
```