Chapter 6: Brain Teasers

**Question 6.1**: You have 20 bottles of pills. 19 bottles have 1.0 gram pills, but one has pills of weight 1.1 grams. Given a scale that provides an exact measurement, how would you find the heavy bottle? You can only use the scale once.

The trick here is to somehow isolate the difference. If we take n pills from bottle number n and sum the weights we can figure out which bottle is the one with 1.1 gram pills.

For example, lets say we have 3 bottles and bottle 3 is the one with 1.1 gram pills.

Then we would take 1 pill from bottle 1, 2 pills from bottle 2, and 3 pills from bottle 3.

1\*1g + 2\*1g + 3\*1.1g = 1g+2g+3.3g = 6.3 grams

If all the bottles were just 1g, we would get 1g+2g+3g = 6g.

So now if we take the difference, 6.3g-6g = 0.3g. Now the difference in weight for the different pills is 0.1g. To figure out which bottle contributed the extra weight, simply divide the difference in total weight with the difference in weight for the different pills: 0.3g / 0.1g = 3. So the third bottle is the one with 1.1g pills.

**Question 6.2**: There is an 8x8 chess board in which two diagonally opposite corners have been cut off. You are given 31 dominoes, and a single domino can cover exactly two squares. Can you use the 31 dominos to cover the entire board? Prove your answer (by providing an example or showing why it’s impossible).

This will be impossible because no matter how you place the domino, since each domino must cover two spots, you will always have one domino taking a spot in another row which would again make that row have an odd number of spot. Even - odd = odd so no matter what, you cannot place dominos in such a way that they all fit.

**Question 6.3**: You have a five-quart jug, a three-quart jug, and an unlimited supply of water (but no measuring cups). How would you come up with exactly four quarts of water? Note that the jugs are oddly shaped, such that filling up exactly “half” of the jug would be impossible.

Step 1: Pour 5 quarts into the five-quart jug.

Step 2: Pour from the five-quart jug into the three-quart jug until the three-quart jug is full. This will leave two-quarts in the five-quart jug.

Step 3: Empty out the three-quart jug

Step 4: Pour the two quarts from the five-quart jug into the three-quart jug.

Step 5: Pour 5 quarts into the five-quart jug

Step 6: Pour from the five-quart jug into the three-quart jug which right now contains 2 quarts. You now have four quarts in the five-quart jug

**Question 6.4:** A bunch of people are living on an island, when a visitor comes with a strange order: all blue-eyed people must leave the island as soon as possible. There will be a flight out at 8:00pm every evening. Each person can see everyone else’s eye color, but they do not know their own (nor is anyone allowed to tell them). Additionally, they do not know how many people have blue eyes, although they do know that at least one person does. How many days will it take the blue-eyed people to leave.

Let’s say n represents the number of number of people with blue eyes.

Case 1 → n = 1:

IF there is only one person with blue eyes, he can look around realize that since no one else has blue eyes, he would know it was him and would leave on that day. Therefore, it took 1 day.

Case 2 → n = 2:

IF there are two people with blue eyes, they would see each other and realize that if n=1, then the one person would have left yesterday, therefore, n must equal two and would therefore take two days.

Case 3 → n>2

Similarly, for n blue-eyed it would take n days

**Question 6.5**: There is a building of 100 floors. If an egg drops from the Nth floor or above, it will break. If it’s dropped from any floor below, it will not break. You’re given two eggs. Find N, while minimizing the number of drops for the worst case.

**Question 6.6**: There are 100 closed lockers in a hallway. A man begins by opening all 100 lockers. Next, he closes every second locker. Then, on his third pass, he toggles every third locker (closes it if it is open or opens it if it is closed). This process continues for 100 passes, such that on each pass i, the man toggles every ith locker. After his 100th pass in the hallway, in which he toggles only locker #100, how many lockers are open.

Chapter 7: Mathematics and Probability

**Question 7.1**: You have a basketball hoop and someone says that you can play one of two games.

Game 1: You get one shot to make the hoop.

Game 2: You get three shots and you have to make two of three shots.

If p is the probability of making a particular shot, for which values of p should you pick one game or the other?

So for Game 1, the probability is simply p.

For Game 2, we will use the following: SUM( (n choose k) \* p^k \* q^(n-k) ) where q = probabilit of not making it, i.e., (1-p)

[(3 choose 3)\* p^3 \* (1-p)^(0) ] + [ (3 choose 2)\* p^2 \* (1-p)^(1) ] = p^3 + 3(p^2)(1-p) =

p^3 + 3(p^2-p^3) = p^3 + 3p^2 - 3p^3 →

3p^2 - 2p^3

Now setting them equal to each other:

3p^2 - 2p^3 = p → 3p^2 - 2p^3 - p = 0 → p(3p - 2p^2 -1) = 0 → 3p - 2p^2 - 1 = 0 → 2p^2 - 3p + 1 = 0 → (2p - 1)(p - 1) = 0

In our case p-1 makes no sense since proabilities are between 0 and 1. Therefore, we will use the (2p-1) = 0 → p = ½. Therefore, if p < ½, we will take game 1, otherwise game 2.

**Question 7.2**: There are three ants on different vertices of a triangle. What is the probability of collision (between any two or all of them) if they start walking on the sides of the triangle? Assume that each ant randomly picks a direction, with either direction being equally likely to be chosen, and that they walk at the same speed. Similarly, find the probability of collision with n ants on an n-vertex polygon.

Notice the only way the ants can avoid collision is if they all chose to go the same direction. Since we are told that the probability of picking a direction is equal, and for any given vertex in a polygon, there are two edges, that makes the probability of choosing a direction ½ or 50%. The two possible directions are either left or right. Because the ant’s decision on which direction to chose is independent from the other ants, we get the following equation which can be generalized to n antson an n-vertex polygon:

2\*(½)^n → 2^1 \* 2^(-n) → 2^(1-n) = the probability of going the same direction

Therefore, the probability of collision is 1 - 2^(1-n)

**Question 7.3**: Given two lines on a Cartesian plane, determine whether the two lines would intersect.

This problem is extremely ambiguous. What we need to clarify with the interviewer what format the lines are in. Are they in slope-intercept y= mx+b form, or are we simply given two points for each line?

Overall, this problem is reduced to the problem of figuring out whether or not the two lines are parallel. By definition of intersecting lines, if the two lines are not parallel, they must intersect at some point.

Pseudocode

//Line Object

public static class Line{

Point p1, p2;

}

public static class Point{

double x, y;

}

public static boolean doTheyIntersect(Line line1, Line line2){

//First we check if the two lines are the same

if(line1 == line2) return true;

double slope1 = calculateSlope(line1);

double slope2 = calculateSlope(line2);

if(slope1 == slope2) return true;

else return false;

}

public static double calculateSlope(Line line){

//(y2-y1)/(x2-x1)

double y2 = line.p2.y;

double y1 = line.p1.y;

double x2 = line.p2.x;

double x1 = line.p1.x;

return (y2-y1)/(x2-x1);

}

**Question 7.5**: Given two squares on a two-dimensional plane, find a line that would cut these two squares in half. Assume that the top and the bottom sides of the squares run parallel to the x-axis.

Once again, there are a lot of things we need to clarify with the interviewer. First, what is the format of the line? Slope-intercept y=mx+b or two points.

Say its the same as in Question 7.3, that is, we are given two points for each point. Now we simply calculate the midpoint mp = (x2-x1)/2.

Now we have the following two points for our new line which will divide the square into half:

Point 1: (mp, y1)

Point 2: (mp, y2)

Thus, we can create a Line Object with these two points and return that as our answer assuming thats the format the answer should be in.

**Question 7.6:** Given a two-dimensional graph with points on it, find a line which passes the most number of points.

We can solve this problem in O(n^2) time. We will use a hashtable where the key is the slope and value is an object that contains one point, and the count representing the number of times we’ve seen that slope. We will iterate through each point and find the slope between that point and the other point. We can then look through our hashtable’s keyset in O(n) time to find the slope with the largest count (we can optimize this by keeping a max reference in O(1))

Recall the value is an object that contains a point which we will use to create Line object.

Pseudocode:

public static class valueObject{

Point p;

int count;

public valueObject(Point p, int count){

this.p = p;

this.count = count;

}

}

public static Line findLineWithLargestCover(Graph g){

HashMap<Double, valueObject> mHM = new HashMap<Double, valueObject>();

for each point p in graph g{

for each point q in graph g{

if(p != q){

double tSlope = calculateSlope(p, q);

if(mHM.get(tSlope) == null){

//add new entry

mHM.put(tSlope, new valueObject(p, 1));

}

else{

mHM.put(tSlope, mHM.get(tSlope).count++); //increment the count

}

}

} //closes if p != q

}

}

Entry<Double, valueObject> mEntry = findMaxCount(mHM);

Line line = new Line();

double tSlope = mEntry.getKey();

double tPoint = mEntry.getPoint().p;

line.p1 = tPoint;

line.p2 = new Point(tPoint.x, tPoint.x \* tSlope + 5); //create random point with same slope

return line;

}//closes function findLineWithLargestCover

**Question 7.7**: Design an algorithm to find the kth number such that the only prime factors are 3, 5, and 7

3^i \* 5^j \* 7\*k

i j k

---------------------

0 0 0

1 0 0

1 1 0

1 1 1

2 0 0

2 1 0

2 1 1

2 2 0

2 2 1

2 2 2

Notice the pattern. Essentially we need to implement a triple nested for loop and keep track of the number of such number produced. When this count == k, return value;

Chapter 8: Object-Oriented Design

When answering any OO-Design Question we need to consider the following:

1. Who, what, where, when, how, why (who will be using this object, why will it be used, when and how, etc…)

2. Define core objects

3. Define relationships between core objects

4. Define Actions the objects need to perform

**Question 8.1**: Design the data structures for a generic deck of cards. Explain how you would subclass the data structures to implement blackjack.

1. This will probably be used as a game by two or more players, a dealer, and must have a way of keeping score

2. The Core Objects will be as follows: Card, Deck, Game (class, which will be extended by specific games), User, Player, Dealer, Score

3. A Deck object is *composed* of Card objects. A specific Game object will implement the root Game class which will be abstract. User Object will be inherited by Player and Dealer. Score object can be a separate class if necessary, or it can simply be part of the Derived Game class.

4. A Card object will store rank and class. A Deck object will hold an array of Card objects and will perform the following actions: shuffle(), getTop(int numCardsToGet), createDeck(int numCards). Game class will store a Deck. Games like Blackjack will extend Game and will implement the logic of the specific game. A User Object will be an abstract class. A Player Object and Dealer Object will implement User and house actions to preformMove(), pickCard(), UpdateScore(). Similarly for Dealer, performMove() will simply be Deal().

**Question 8.2:** Imagine you have a call center with three levels of employees: respondent, manager, and director. An incoming telephone call must be first allocated to a respondent who is free. If the respondent can’t handle the call, he or she must escalate the call to the manager. If the manager is not free or not able to handle it, then the call should be escalated to a director. Design the classes and data structures for this problem. Implement a method dispatchCall() which assigns a call to the first available employee.

1. This will be used in a call center with multiple employees. Depending on the size of the employees we will use a data structure to hold them accordingly. For example, if there is a large number of employees, we will use hashmaps (one for each level)

2. The core objects are Employee (which will be extended by respondent, manager, and director), Call object, and a Router object

3. Employee Object will hold basic information such as employee id, name, rank. The classes that extend it will have the logic associated with each level of employee. A Call object will hold information like caller, rank of call, etc. The Router object will manage the escalation of a Call across the levels accordingly. The Router object will be singleton. It will be composed of all the Employees

public class Router{

private static Router instance;

private ArrayList<HashMap<Employee>> mEmployees; //ArrayList of hashmaps for each level of employee

private Queue mQueue;

//make singleton

public static Router(){

if(instance == null) return (instance = new Router());

else return instance;

}

public void dispatchCall(Call tCall){

if(mQueue not empty){

enqueue tCall

set tCall to mQueue.dequeue();

}

int rank = tCall.getRank(); //Lets say rank is 0 for respondent, 1 for manager, and 2 for dir

if(rank > 2) mQueue.enqueue(tCall); return;

mHashMap = mEmployees.get(rank);

if no employee is available, call.escalateRank(); disPatchCall(tCall);

else connect(tCall, employee);

}

}

**Question 8.3:** Design a musical jukebox using object-oriented principles

1. Who will be using Jukebox? A small group of people in a small bar (i.e. couple albums) or a machine that is capable of playing multiple tracks at the same time in a distributed fashion. This is where you would discuss whether Jukebox will singleton or factory.

2. The core objects will be as follows: Player, Format, Album, Track

3. An Album will consists of an array of Track objects. A player object will consist of a Format and an array of albums.

4. A track object will house features such as artist, track number, length, album, album art, year, etc. An album object will consist of multiple Tracks and will house features such as getInfo(). A player object will perform actions such as Play(), Pause(), NextTrack(), PrevTrack(), NextAlbum(), PrevAlbum().

**Question 8.4:** Design a parking lot using object-oriented principles.

1. Who will be using this parking lot, where, when,etc… For example, if a lot of people will be using it, we discuss what data structures to use. Is this valet parking? Is this in a building or underground?

2. The core objects in this should be simple. We will have a ParkingLot Object, Space Object, Vehicle Object.

3. A ParkingLot Object will be composed of Space Objects. A Vehicle will have one Space Object to which it is occupying. A Space object will have an id and boolean to represent whether or not it is being occupied.

4. The ParkingLot object can potentially be a bit more smart than a dumb container object. IT can act as a router as well which will provide mechanism to find an available spot. It will have functions such as requestSpot(), findAvailableSpots(), count, etc…

**Chapter 10: Scalability and Memory Limits**

**Question 10.1**: Imagine you are building some sort of service that will be called by up to 1000 client applications to get simple end-of-day stock price information (open, close, high, low). You may assume that you already have the data, and you can store it in any format you wish. How would you design the client-facing service which provides the information to client applications? You are responsible for the development, rollout, and ongoing monitoring and maintenance of the feed. Describe the different methods you considered and why you would recommend your approach. Your service can use any technologies you wish, and can distribute the information to the client applications in any mechanism you choose.

Assumptions: Client will visit website and query for a specific stock(s)

What we need to consider:

- Client usability

- Scalability

What we will need

SQL Database, Application Server, Web Frontend, Load Balancer

Example: SQL Server ←→ Hibernate ←→ JBoss Application Server ←→ Spring Framework ← → JSP/JavaScript/AngularJS/DataTables

I can explain further… Very Similar technology stack to what I am doing now.

**Question 10.2**: How would you design the data structures for a very large social network like Facebook or LinkedIn? Describe how you would design an algorithm to show the connection, or path, between two people (e.g., Me → Bob → Susan → Jason → You).

This will be stored as a Graph. A graph consists of vertices and edges. A vertex can have multiple edges. In terms of Data Structures, an Edge will be an Object with two Vertices. A Vertex will be a user with all their information stored in some HashMap. To traverse, we will use some complex variant of the common graph traversal algorithms: DFS and BFS. Note that in order to find a connection, we are more likely to be local (i.e. few hops); therefore, we will use BFS (go into discussion on difference between DFS and BFS and why BFS for social networks is more efficient). Next, we can further store graphs based on location; in other words, a facebook user in the U.S. is more likely to be connected to another facebook user in the U.S. rather than some user in China. Consequently, we can have a localized cluster (which can further be divided into sub clusters) in the U.S. and one for China. In BFS, the marking of nodes as visited is not a good idea because there could multiple searches. Instead, for every search query we can create an object with a hashmap to mimic the marking of nodes. Again, we must deal with scalability and reliability. Discuss Load Balancers, Replication, Fail-Safe server farms…

**Question 10.3**: Given an input file with four billion non-negative integers, provide an algorithm to generate an integer which is not contained in the file. Assume you have 1 GB of memory available for this task.

FOLLOW UP: What if you have only 10MB of memory? Assume that all the values are distinct and we now have no more than one billion non-negative integers.

1 GB = 1024 MB = 2^10 MB. 1 MB = 2^20 bytes. So 1 GB = 2^10 MB \* (2^20 bytes)/(1 MB) = 2^30 bytes.

8 bits in a byte → 2^30 \* 2^3 bits / 1 byte = 2^33 bits = ~ 8 billion bits.

So, if we are allowed 1 GB of memory, we can essentially use a bit array to denote whether or not we have seen the number. We will iterate once through the input file, for every number we see, we mark the bit at the corresponding index as 1. Then we will iterate through the bit array and find which ones are still set to 0 to indicate the numbers that are not in the file.

Follow Up: If we are only given 10 MB of memory, we can do the following.

10MB = 10\*1024 bytes = 10240 bytes \* 8bits/byte = 81920 bits.

We can therefore can perform the above algorithm except in section/grouping. The first group we will scan only for numbers between 1-80000.Then, the second group will be 80001-160000, etc. For each section we will iterate through the bit array and determine which numbers within that section are missing.

**Question 10.4**: You have an array with all the numbers from 1 to N, where N is at most 32,000. The array may have duplicate entries and you do not know what N is. With only 4 kilobytes of memory available, how would you print all duplicate elements in the array?

4KB = 2^2 KB \* 2^10 bytes/KB = 2^12 bytes \* 2^3 bits/byte = 2^15 bits = ~ 32000.

Once again we can use a bit array to determine the duplicate number. We scan through the array of numbers, and for every number we see we find the bit at the corresponding index. If it is 0, set it to 1. If it is one, we have found a duplicate entry.

**Question 10.5**: If you were designing a web crawler, how would you avoid getting into infinite loops?

Obviously we can use a hash table to indicate whether or not we visited a page; however, the real question is how to we distinguish between two pages. We cannot necessarily do it by url because a website can have dynamic content on a page that both have the same url. Similarly, we cannot do it by content either because a page can have small sections which are dynamic but the overall page is the same.

Therefore, we must use a hybrid, which would avoid getting caught in cycles. We will use a ranking system where prioritize a page to be indexed.

**Question 10.6**: You have 10 billion URLs. How do you detect the duplicate documents? In this case, assume that “duplicate” means that the URLs are identical.

10 billion urls would at about 100 chars per url is ~1 trillion bytes which is about 900GB (assuming 1 char is 1 byte). Obviously no one system would have 900GB of memory, so we have two options:

Option 1:

Use a distributed system to process the URLs (using a hashmap).

Option 2:

Use the hard-drive and a file based database/hashmap.

**Question 10.7:** Imagine a web server for a simplified search engine. This system has 100 machines to respond to search queries, which may then call out using processSearch(String query) to another cluster of machines to actually get the result. The machine which responds to a given query is chosen at random, so you can not guarantee that the same machine will always respond to the same request. The method processSearch is very expensive. Design a caching mechanism to cache the results of the most recent queries. Be sure to explain how you would update the cache when data changes.

First we must consider whether or not want an MRU or a LRU cache. Since this if for serving up web pages, we would want to use a LRU (Least Recently Used) cache instead of a MRU (Most Recently Used) cache; that is, when it comes time to replace a cache slot, we remove the least recently used one.

The data structure we can use to implement this is a LinkedList which would allow for easy (O(1)) deletion and insertion.

Obviously, for scalability purposes we would want to distribute this over several machines. We can do this in the following ways:

Option 1: Each machine has its own cache

Option 2: Each machine is a replication

Option 3: The cache itself is segmented/distributed over the machines

Option 1 Pros: Easy to set up and manage, inexpensive since there are no machine to machine calls.

Cons: Since we cannot determine which machine would get hit, this would not be an effective cache.

Option 2 Pros: Effective cache

Cons: Expensive to manage the replicated state

Option 3 Pros: Effective cache, inexpensive search

We would most likely go with Option 3 and use some sort of hashing function to determine on which cache machine the content would be on.

**Chapter 12: Testing**

The main questions we need to ask when answering a testing question are:

1. Who will use it? And why?

2. What are the use cases?

3. What are the bounds of use?

4. What are the stress/failure conditions?

5. How would you perform the testing?

Black Box Testing vs White Box Testing: Black box is when you are given software as is; white box is when you additional programmatic access to test individual functions

When testing a function:

1. Normal Case

2. Extreme Cases

3. Nulls and illegal input

4. Strange input

**Question 12.1:** Find the mistake(s) in the following code:

1. unsigned int i;
2. for(i = 100; i>=0; --i)
3. printf("%d**\n**", i);

First off, what is the intended purpose of this code? He is trying to print all the numbers from 100 to 0 (counting down).

Notice that the variable i is an unsigned int. This means i can never be negative and will always be >=0. Therefore, this will cause an infinite loop. Second, the %d should be %u for unsigned int.

**Question 12.2**: You are given the source to an application which crashes when it is run. After running it ten times in a debugger, you find it never crashes in the same place. The application is single threaded, and uses the C standard library. What programming errors could be causing this crash? How would you test each one?

The most important thing to discuss here is what is the application used for, who is using it, and the exact steps necessary to reproduce the error. However, note that since we are given that it does not crash at the sample place, and it is a single threaded application, we know that the cause must be something along the lines of:

1. Random variable

2. Uninitialized variables

3. Memory leaks

**Question 12.3:** We have the following method used in a chess game: boolean canMoveTo(int x, int y). This method is part of the Piece class and return whether or not the piece can move to position (x,y). Explain how you would test this method.

Recall that when testing a function, we need to focus on the following:

1. normal case,

2. extreme case

3. null and illegal input

4. strange input

Lets assume that this is a typical 8x8 chessboard. Therefore we know 0<=x<=7 and 0<=y<=7

So The normal case would be testing x,y for any input combination that falls within those ranges.

The extreme case is to test with negative numbers or numbers that fall outside the bounds.

You cannot pass null or illegal input, otherwise we would have a compile time error.

**Question 12.4:** How would you load test a webpage without using any test tools?

Assumption: When it says we cannot use any test tools, it is referring to pre-built software like JMeter.

The things we need to test for “load testing” are:

1. response time

2. throughput (i.e. rate of successfully sent packets)

3. Max load

We can do this by building an application that measure each of the above. To simulate users is the tricky question: We can build a multithreaded application where each thread can simulate a set of users, but once again if they are on the same machine, we are not really testing the real use case. We may even create an application that runs on a distributed system to better represent real world scenarios.

**Question 12.5:** How would you test a pen?

Once again, we fail to our set of questions we need to answer:

1. Who will use it and why?

2. How will it be used?

3. What are the bounds of use

4. What are stress/failure

Answers: This product will most likely be used by a human with a hand to write something. It will be used probably on paper and in a relatively dry environment. Thus, we test the following:

1. Does the ink flow smoothly or in an uneven rate

2. Does the pen write on intended items (paper)

3. Is the pen supposed to be water resistant? If so, is the ink actually water resistant?

4. Is the pen non-toxic?

5. Unintended Use/Stress Testing: What happens when you step on the pen, does it break? About in extreme temperatures, does it still write? What happens if you throw it, does it leak?

**Question 12.6:** How would you test an ATM in a distributed banking system?

The key statement here is distributed. So, by far, the most important thing we need to test is to make sure transactions are thread-safe (aka atomic), meaning we test whether or not the locking system in place actually works. This is a heavy distributed systems questions: relate to ACID principles, Atomic, Consistent, Isolated, Durable

We need to test the following types of transactions: withdrawal, deposit, transfer, even read-only operations like just view balance

**Chapter 14: Java Knowledge Based**

**Question 14.1**: In terms of inheritance, what is the effect of keeping a constructor private?

By making the constructor private, this means that no other class can directly instantiate this class. Instead, they’d have to rely on indirect instantiation. For example say we have an error logger class with a private constructor. Maybe, it may be more efficient to instantiate this class on an error (i.e. there’s a public static function the ErrorLogging class named logError which would instantiate the ErrorLogging accordingly).

**Question 14.2**: In Java, does the finally block get executed if we insert a return statement inside the try block of a try-catch-finally?

The finally block is always executed after the try block exits. Even when there is an exception or a return statement, the finally block will get executed. It is good practice to put in type of clean-up code in finally block. The only way a finally block is not executed is if JVM unexpectedly crashes/gets killed.

**Question 14.3**: What is the difference between final, finally, finalize?

final is the equivalent to a const variable/method/class. i.e. this variable/method/class cannot change during the course of runtime.

finally is part of the try/catch/finally block. The finally block is always executed after the try block exits. (Read answer to Question 14.2)

finalize is a function called by the garbage collector when JVM has determined that no more references to an object exist.

**Question 14.5:** Explain what object reflection is in Java and why it is useful.

Java object reflection is a way of looking at a class through code (at runtime) and figuring out if it has a method or field and then be able to call that method/field accordingly. We use object reflection in the following ways:

1. observing/debugging runtime behavior

2. Say we have a packaged class whose source code we cannot view. This would be a situation where using object reflection at runtime maybe crucial

Note that reflection is slow, so it should probably never be used in production quality code

**Question 14.6**: Implement a CircularArray class that supports an array-like data structure which can be efficiently rotated. The class should use a generic type, and should support iteration via the standard for (Obj o : circularArray) notation

1. **public** **class** CircularArray<T>{
2. *//We will use the generic type T*
3. **private** T[] elements;
4. **private** **int** head = 0;
6. **public** CircularArray(**int** size){
7. items = (T[])**new** Object[size];
8. }
10. **public** **void** rotate(**int** shiftRight){
11. **if**(shiftRight < 0){
12. shiftRight += elements.length;
13. }
14. head = (head + shiftRight) % elements.length;
15. }
17. **public** T get(**int** i){
18. **if**(i<0 || i>=elements.length){
19. *//throw exception*
20. **return** **null**;
21. }
22. **int** theIndex = (head + i)%elements.length;
23. **return** elements[theIndex];
24. }
26. **public** **void** set(**int** i, T element){
27. **if**(i<0 || i>=elements.length){
28. *//throw exception*
29. **return** **null**;
30. }
31. **int** theIndex = (head + i)%elements.length;
32. elements[theIndex] = element;
33. }
35. }

**Chapter 15:** **Databases**

**Question 15.1**: Write a SQL query to get a list of tenants who are renting more than one apartment.

select \* from Tenants inner join (select TenantID from AptTenants group by TenantID having count(\*) > 1) x on Tenants.TenantID = x.TenantID

**Question 15.2:** Write a SQL query to get a list of all buildings and the number of open requests (Requests in which status equals ‘Open’).

select \* from Buildings left join Apartments on Buildings.BuildingID = Apartments.BuildingID left join Requests on Apartments.AptID=Requests.AptID where Requests.Status=’Open’;

**Question 15.3:** Building #11 is undergoing major renovation. Implement a query to close all requests from apartments in this building.

update Requests set status = ‘Closed’ where AptID in

(select AptID from Apartments where BuildingId = 11)

**Question 15.4:** What are the different types of joins? Please explain how they differ and why certain types are better in certain situations.

The types of joins are as follows:

Inner Join: Only contains intersection of two tables

Outer Join

- Left Outer Join: Contains intersection and rows from left table

- Right Outer Join: Contains intersection and rows from right table

- Full Outer Join: Left + Right

**Question 15.5:** What is denormalization? Explain the pros and cons.

Denormalization is the process of adding extra redundant information in tales so as to avoid costly join operations. Pros: Dont have to do costly join operation; Cons: Takes more space

**Chapter 16: Threads and Locks**

**Question 16.1**: What is the difference between a thread and a process?

Process is an instance of a program in execution. Threads exist within processes.

The differences between a thread and a process are as follows:

1. Processes run in their own memory space; threads run in shared memory spaces of the parent process

2. Threads can communicate directly with each other; Process must use some form of IPC (pipes, fds, sockets,...)

3. Threads are launched using pthread.create(); processes are launched by fork()

**Question 16.2**: How would you measure the time spent in a context switch?

A context switch is when the OS decides to allocate cpu resources from one process to another. To measure the time spent in a context switch is slightly difficult given standard/normal operation. This is because the OS implements its own scheduling algorithm and we can never really be sure when the OS will decide to context switch; consequently, it would be very difficult to know where to efficiently put timestamps. Instead, we will force p2 to run exactly after p1 by using a pipe (i.e. eliminating multitasking). So on the last instruction of p1, we would record the timestamp t1 and on the first instruction of p2, we would record the timestamp t2. Thus, the context switch is nothing bu t2-t1

**Question 16.3:** In the famous dining philosophers problem, a bunch of philosophers are sitting around a circular table with one chopstick between each of them. A philosopher needs both chopsticks to eat, and always picks up the left chopstick before the right one. A deadlock could potentially occur if all the philosophers reached for the left chopstick at the same time. Using threads and locks, implement a simulation of the dining philosophers problem that prevents deadlocks.

Recall that there are four necessary condition to a deadlock:

1. Mutual exclusion

2. Hold and Wait

3. No Preemption

4. Circular Wait

IF we can eliminate any of the above, we can remove the deadlock. For example, let us eliminate Hold and Wait, meaning if the dining philosopher is unable to pick up his right chopstick, he will put down (i.e. let go of the left chopstick).

**Chapter 17: Moderate**

**Question 17.1:** Write a function to swap a number in place (that is, without temporary variables).

This is a classic xor question (same trick we use when we want to reverse a string in O(logn) time and in place.

To swap the values of x and y we do the following:

x^=y

y^=x

x^=y

**Question 17.2:**  Design an algorithm to figure out if someone has won a game of tic-tac-toe.

Assumptions: 3x3 board, this algorithm will be called after each move

Remember that a winner in tic-tac-toe is one who has three in a row (either horizontal, vertical, or diagonal)

Let this be how the board is organized

|  |  |  |
| --- | --- | --- |
| (0,0) | (0,1) | (0,2) |
| (1,0) | (1,1) | (1,2) |
| (2,0) | (2,1) | (2,2) |

1. **public** Boolean didWin(Player p, Board b){
2. Boolean retVal = **false**;
3. **for**(Point pnt : Board.getPoints()){
4. **if**(pnt.getMarker() == p.getMarker()){
5. Boolean retVal = explore(pnt, b);
6. **if**(retVal) **return** **true**;
7. }
8. }
9. **return** **false**;
10. }

13. **public** Boolean explore(Point pnt, Board b){
14. **int** matchCount = 0;
16. *//First we explore horizontal*
17. **for**(**int** x=0; x<3; x++){
18. **if**(b.getPoint(x, pnt.getY()).getMarker() != pnt.getMarker())
19. **break**;
20. **else** matchCount++;
21. }
22. **if**(matchCount == 3) **return** **true**;
23. **else** matchCount = 0;
25. *//Then we explore vertical*
26. **for**(**int** y=0; y<3; y++){
27. **if**(b.getPoint(pnt.getX(), y).getMarker() != pnt.getMarker())
28. **break**;
29. **else** matchCount++;
30. }
31. **if**(matchCount == 3) **return** **true**;
32. **else** matchCount = 0;
34. *//Then we explore diagonal*
35. **for**(**int** d = 0; d<3; d++){
36. **if**(b.getPoint(d,d).getMarker() != pnt.getMarker())
37. **break**;
38. **else** matchCount++;
39. }
40. **if**(matchCount == 3) **return** **true**;
42. **return** **false**;
43. }

Note that the above solution can easily be extended to an NxN board, not just a 3x3 board

**Question 17.3:**  Write an algorithm which computes the number of trailing zeros in n factorial

The obvious solution would be to simply compute n factorial, convert it to a string, and then count the number of zeros at the end of the string. However, there must be a more efficient solution which we can develop by looking at the properties of n!

Notice that the only way we add a trailing zero is by multiplying by 10. So everytime we have a 2x5 we add a trailing zero. Therefore, given N, we simply need to count the number of 2s and 5s between 1 and N inclusive.

1. **public** **static** **int** calculateNumberOfTrailingZerosInFactorial(**int** N){
2. **int** countTwo = 0, countFive=0;
3. **for**(**int** i = 1; i<=N; i++){
4. **if**(i%2==0){
5. countTwo++;
6. **int** temp = i;
7. **while**((temp/2)%2 == 0){
8. countTwo++;
9. temp/=2;
10. }
11. }
12. **if**(i%5==0){
13. countFive++;
14. **int** temp = i;
15. **while**((temp/5)%5 == 0){
16. countFive++;
17. temp/=5;
18. }
19. }
20. }
22. **return** Math.min(countTwo, countFive);
23. }

**Question 17.4:** Write a method which finds the maximum of two numbers. You should not use if-else or any other comparison operator.

Ultimately, for problem like this we need to use binary operators. But before that lets examine what it means to find the max.

Lets say we have two numbers x and y. Originally, we know that if(x>y) return x else return y.

We need to continually change/reword this problem until we get to the final answer.

Notice that if (x-y) is negative, we know y is greater. Lets say we have another temp variable that will be 1 if (x-y) is negative and 0 if (x-y) is positive.

int temp = (x-y);

if(temp < 0) temp = 1

else temp = 0;

return x - temp \*(x-y);

notice that this will correctly return the max value, but we still have an if clause. This is where binary come in. Recall that modern computers store negative numbers using the twos complement (meaning the most significant bit) is 1 if negative.

1. **public** **int** MAX(**int** x, **int** y){
2. **int** temp = (x-y);
3. temp = (temp >> 31) & 1; *//accordingly set to 0 or 1*
4. **return** x - temp\*(x-y);
5. }

**Question 17.6:** Given an array of integers, write a method to find indices m and n such that if you sorted elements m through n, the entire array would be sorted. Minimize n - m (that is, find the smallest such sequence).

1 2 4 7 10 11 7 12 6 7 16 18 19

find divergent point (aka point where its no longer increasing). This is at the second 7. Then we scan the array again until we find a value that is greater than this divergent point. This is at 10. This is m.

Then we traverse from the right (making sure it is sorted) until we find either the second divergent point or a point where it is less than m. This will be point n.

**Question 17.7:** Given any integer, print an English phrase that describes the integer (e.g., “One Thousand, Two Hundred Thirty Four”).

We cannot go to infinity, so we need to find the lower and upper bounds.

In java, int is always 32-bit which means the largest number possible is 2^32 -1 = 4,294,967,295 or ~4 billion. The lower bound is the same, but negative.

1. **public** **static** **final** String[] ones = {“Ones”, “Two”, “Three”, “Four”, “Five”, “Six”, “Seven”, “Eight”, “Nine”};
2. **public** **static** **final** String[] teens = {...};
3. **public** **static** **final** String[] tens = {...};
4. **public** **static** **final** String[] rest = {“Hundred”, “Thousand”, “Million”, “Billion”};

7. **public** **static** string numToString(**int** number){
8. **if**(number == 0) **return** "Zero";
9. **if**(number < 0) **return** "Negative " + numToString(-1 \* number);
11. StringBuilder mSB = **new** StringBuilder();
12. mSB.append("");
13. **while**(number > 0){
14. **if**(number % 1000 != 0){
15. *//meaning we convert for number less than 1000*
16. *//hundreds place*
17. **if**(number >= 100){
18. mSB.append("" + ones[number / 100 - 1] + rest[0] + " ");
19. number %= 100;
20. }
22. **if**(number >=11 && number <=19){
23. mSB.append("" + tens[number - 11] + " ");
24. }
25. **else** **if**(number == 100 || number >=20){
26. mSB.append("" + ones[number -1] +" ");
27. }
28. }
29. number /= 1000;
30. }
32. **return** mSB.toString();
33. }

**Question 17.8:** You are given an array of integers (both positive and negative). Find the contiguous sequence with the largest sum. Return the sum.

1. **public** **static** **void** LISum(Integer[] mArray){
2. *//This will implement the Longest Increasing Sum algorithm*
3. **int** maxSum = 0;
4. **int** sum = 0;
5. **for**(**int** i=0; i<mArray.length; i++){
6. sum += mArray[i];
7. **if**(maxSum < sum){
8. maxSum = sum;
9. }
10. **else** **if**(sum < 0){
11. sum = 0;
12. }
13. }
14. **return** maxSum;
15. }

**Question 17.12**: Design an algorithm to find all pairs of integers within an array which sum to a specified value.

An O(n) solution is to iterate through the array and put all the values into a hashmap. Then iterate through the hashmap’s keyset checking the difference between sum value and the current value exists. If so, print/add to the return structure.

**Chapter 18:** Hard

**Question 18.1:** Write a function that adds two numbers. You should not use + or any arithmetic operators.

Obviously, we have to deal with binary operators.

public int add(int a, int b){

if(b==0) return a;

int sum\_without\_carry = a ^ b;

int carry = ( a&b ) << 1;

return add(sum\_without\_carry, carry);

}

**Question 18.2:** Write a method to shuffle a deck of cards. It must be a perfect shuffle -- in other words, each of the 52! permutations of the deck has to be equally likely. Assume that you are given a random number generator which is perfect.

Essentially at each iteration between 1 to n, we shuffle n-1 and then using the random generator randomly swap the nth card. This makes each of the 52! permutations equally likely.

public void shuffleDeck(int[] cards){

for(int i = 0; i<cards.length; i++){

int random = rand(0, i);

//swap

int temp = cards[i];

cards[i] = cards[random];

cards[random] = temp;

}

}

**Question 18.3:** Write a method to randomly generate a set of m integers from an array of size n. Each element must have equal probability of being chosen.

public int[] generateSet(int m, int[] arrayN){

int[] retSet = new int[m];

for(int i =0; i<m; i++){

retSet[i] = arrayN[i];

}

for(int i = m; i < arrayN.length; i++){

int random = rand(0, i);

if(random < m){

retSet[random] = arrayN[i];

}

}

return retSet;

}

**Question 18.4**: Write a method to count the number of 2s between 0 and n

Brute force would be to iterate through 0 to n while converting each number into a string and then iterating the string and counting the number of 2s.

Better solution: notice that

0-9 1

10-19 1

20-29 11

30-39 1

40-49 1

50-59 1

60-69 1

70-79 1

80-89 1

90-99 1

**Question 18.5:** You have a large text file containing words. Given any two words, find the shortest distance (in terms of number of words) between them in the file.

public int shortestDistanceBetweenWords(String[] words, String w1, String w2){

int w1pos = -1, w2pos = -1;

int minD = Integer.MAX\_VALUE;

for(int i = 0; i<words.length; i++){

if(words[i].equals(w1){

w1Pos = i;

int dist = w1pos - w2pos;

if(w2pos != -1 && dist < minD){

minD = dist;

}

}

else if(words[i].equals(w2){

w2pos = i;

int dist = w2pos - w1pos;

if(w1pos != -1 && dist < minD){

minD = dist;

}

}

}

return minD;

}

**Question 18.6:** Describe an algorithm to find the smallest one million numbers in one billion numbers. Assume that the compute memory can hold all one billion numbers.

If we know the range or numbers, and it is below 1 billion (or some amount the computer memory can hold), we can use radix/bucket sort to solve this in O(n). However, since we cannot make that assumption, we can solve this in O(nlogn) time using a sorting algorithm like merge sort.

**Question 18.9:** Numbers are randomly generated and passed to a method. Write a program to find and maintain the median value as new values are generated.

We can use a min heap and a max heap. The max heap will be for numbers less than the median whereas the min heap will be for numbers greater than the median. If maxHeap.size() > minHeap.size() then maxHeap.top() will be the median. IF the maxHeap.size() == minHeap.size(), then the average of the tops of the heap will be the median. This algorithm runs on O(logn) time.

**Question 18.10:** Given two words of equal length that are in a dictionary, write a method to transform one word into another word by changing only one letter at a time. The new word you get in each step must be in the dictionary.