Dear All ,

Now days lots of questions and programs has been asked on Java concurrency . Please give more and more time on this topics. Considering these thing in mind we will be sharing you more and more document on this in coming days…!!!

Java Concurrency – Thread Pools.

One of the most generally useful concurrency enhancements delivered in Java 1.5 was the introduction of customizable thread pools. These thread pools give you quite a bit of control over things such as number of threads, reuse of threads, scheduling and thread construction. Let’s review these.

First, thread pools. Let’s dive right into **java.util.concurrent.ExecutorService**, which provides us the basic interface for a thread pool. All thread pools allow submitting **Callable** or **Runnable** instances for future execution. They also provide various pool management methods.

**Pool Management**

Various management methods exist for the pools. You can **shutdown()** the pool, which will reject any future submissions but complete processing of in-process executions and even those that had not yet started but were submitted before the shutdown was initiated. You can also more aggressively perform a shutdownNow(). This will also prevent any future submissions, but it has a few different, notable behaviours. It will not start execution of submitted but unstarted tasks. They will be in the returned list. It will also attempt to stop, or more precisely, **Thread.interrupt()** currently executing tasks. This is a best effort with no guarantee that these tasks will be successfully interrupted.

**ThreadFactory**

In a moment we will get into the **java.util.concurrent.Executors** builder class which can create various thread pool configurations, but first let’s focus for a second on using ThreadFactory. You’ll want to take advantage of **ThreadFactory** support in **Executors** and be in the habit of providing your own. The default **ThreadFactory** will give you give you an incrementing numbered pool naming scheme which is not all that helpful in logs or other monitoring. For the first pool created you’ll get threads named *pool-1-thread-1*, *pool-1-thread-2* and the second one starts with *pool-2-thread-1*, etc. By providing your own **ThreadFactory**, you can have threads named like *ReportProcessingThread1* and *HttpThread1*. Here’s a simple example:

private AtomicLong counter = new AtomicLong();

private String name;

public Thread newThread(Runnable r) {

                    Thread t = new Thread(r);

                    t.setName(name + counter.incrementAndGet());

                    return t;

}

**ThreadFactory** will only be called when a new **Thread** is created. Given that the JDK thread pools will reuse threads whenever possible, this class cannot be used to manage the beginning of execution.

**Executors Builder Methods**

Now back to the **Executors** utility builder methods. They are:

* **newCachedThreadPool()** will give you a thread pool that will reuse threads when possible, creating new ones as needed with no configured limit.
* **newFixedThreadPool(int nThreads)** will give you a thread pool that will use only up to the number of threads specified but will accept as many tasks as submitted for execution running them in submission order.
* **newScheduledThreadPool(int corePoolSize)** is used specifically for scheduling threads with delayed execution, on a recurring schedule on with recurring delay. The returned thread pool implements **ScheduledExecutorService** which exposes the additional scheduling methods **schedule(Runnable command, long delay, TimeUnit unit)**, **scheduleAtFixedRate(Runnable command, long initialDelay, long period, TimeUnit unit)** and **scheduleWithFixedDelay(Runnable command, long initialDelay, long delay, TimeUnit unit)**.
* **newSingleThreadExecutor()** and **newSingleThreadScheduledExecutor()**. These impose no limit on the number of tasks that can be submitted, only ensuring that a single thread/task is executing at a time.

Finally, there are a few helper methods for creating **Callable** instances from **Runnable**. This gets us into the newly created constructs for allowing threads to throw **Exceptions** and return values, something we had to work around quite painfully before

* Sandeep Palo
* Radhika Danda
* Nidhin Chacko

Please try to go thu all the assignment asap. Request you to prepare it in quick span of time  as you are new in the grp so you have to cover lot many things in coming few days . We are looking for your hard work and co-operation .

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, October 06, 2015 4:43 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Rajnish K Tiwari; Sandeep Palo; Radhika Danda; Nidhin Chacko  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Shrijith Shirnali; Sharad Khanna  
**Subject:** RE: Logical and analytical assignment 10( Design Patterns )-Very imp

Adding Nidhin Chacho.

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, September 30, 2015 5:56 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Rajnish K Tiwari; Sandeep Palo; Radhika Danda  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 10( Design Patterns )-Very imp   
**Importance:** High

Dear All ,

Please go thu below materials on Facade design pattern (Please ignore who has cleared the GS interview)

What is the Facade Pattern

The Facade Pattern makes a complex interface easier to use, using a Facade class. **The Facade Pattern provides a unified interface to a set of interface in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use**.

The Facade unifies the complex low-level interfaces of a subsystem in-order to provide a simple way to access that interface. It just provides a layer to the complex interfaces of the sub-system which makes it easier to use.

The Facade do not encapsulate the subsystem classes or interfaces; it just provides a simplified interface to their functionality. A client can access these classes directly. It still exposes the full functionality of the system for the clients who may need it.

A Facade is not just only able to simplify an interface, but it also decouples a client from a subsystem. It adheres to the Principle of Least Knowledge, which avoids tight coupling between the client and the subsystem. This provides flexibility: suppose in the above problem, the company wants to add some more steps to start or stop the Schedule Server, that have their own different interfaces. If you coded your client code to the facade rather than the subsystem, your client code doesn’t need to be change, just the facade required to be changed, that’s would be delivered with a new version to the client.

Clients communicate with the subsystem by sending requests to Facade, which forwards them to the appropriate subsystem object(s). Although the subsystem objects perform the actual work, the facade may have to do the work of its own to translate its interface to subsystem interfaces. Clients that use the facade don’t have to access its subsystem objects directly.

Please note that, a Façade same as an adapter can wrap multiple classes, but a facade is used to an interface to simplify the use of the complex interface, whereas, an adapter is used to convert the interface to an interface the client expects.

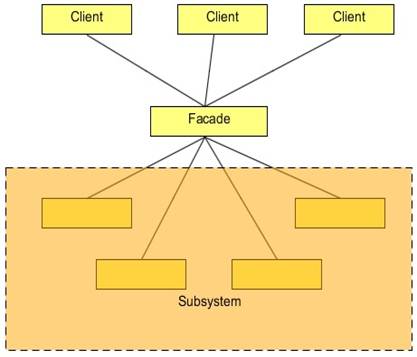
[](http://a3ab771892fd198a96736e50.javacodegeeks.netdna-cdn.com/wp-content/uploads/2015/09/facade_pattern.jpg)

Figure 1

2. Solution to the problem

The problem faced by the clients in using the Schedule Server is the complexity brought by the server in order to start and stop its services. The client wants a simple way to do it. The following is the code that clients required to write to start and stop the server.

View source prints

|  |  |
| --- | --- |
| 1 | ScheduleServer scheduleServer = new ScheduleServer(); |

To start the server, the client needs to create an object of the ScheduleServer class and then need to call the below methods in the sequence to start and initialize the server.

View source prints

|  |  |
| --- | --- |
| 1 | scheduleServer.startBooting(); |
| 2 | scheduleServer.readSystemConfigFile(); |

|  |  |
| --- | --- |
| 3 | scheduleServer.init(); |
| 4 | scheduleServer.initializeContext(); |

|  |  |
| --- | --- |
| 5 | scheduleServer.initializeListeners(); |
| 6 | scheduleServer.createSystemObjects(); |

|  |  |
| --- | --- |
| 7 |  |
| 8 | System.out.println("Start working......"); |

|  |  |
| --- | --- |
| 9 | System.out.println("After work done........."); |

To stop the server, the client needs to call the following methods in the same sequence.

View source prints

|  |  |
| --- | --- |
| 1 | scheduleServer.releaseProcesses(); |
| 2 | scheduleServer.destory(); |

|  |  |
| --- | --- |
| 3 | scheduleServer.destroySystemObjects(); |
| 4 | scheduleServer.destoryListeners(); |

|  |  |
| --- | --- |
| 5 | scheduleServer.destoryContext(); |
| 6 | scheduleServer.shutdown(); |

This looks a burden to them, they are not interested in doing all these stuffs, and why would they? Even though this might look interesting to some of the clients who might be interested in the low-level interface of the system, most of them disliked it.

To resolve this, we will create a facade class which will wrap a server object. This class will provide simple interfaces (methods) for the client. These interfaces internally will call the methods on the server object. Let us first see the code and then will discuss more about it.

View source prints

|  |  |
| --- | --- |
| 01 | package com.javacodegeeks.patterns.facadepattern; |
| 02 |  |

|  |  |
| --- | --- |
| 03 | public class ScheduleServerFacade { |
| 04 |  |

|  |  |
| --- | --- |
| 05 | private final ScheduleServer scheduleServer; |
| 06 |  |

|  |  |
| --- | --- |
| 07 | public ScheduleServerFacade(ScheduleServer scheduleServer){ |
| 08 | this.scheduleServer = scheduleServer; |

|  |  |
| --- | --- |
| 09 | } |
| 10 |  |

|  |  |
| --- | --- |
| 11 | public void startServer(){ |
| 12 |  |

|  |  |
| --- | --- |
| 13 | scheduleServer.startBooting(); |
| 14 | scheduleServer.readSystemConfigFile(); |

|  |  |
| --- | --- |
| 15 | scheduleServer.init(); |
| 16 | scheduleServer.initializeContext(); |

|  |  |
| --- | --- |
| 17 | scheduleServer.initializeListeners(); |
| 18 | scheduleServer.createSystemObjects(); |

|  |  |
| --- | --- |
| 19 | } |
| 20 |  |

|  |  |
| --- | --- |
| 21 | public void stopServer(){ |
| 22 |  |

|  |  |
| --- | --- |
| 23 | scheduleServer.releaseProcesses(); |
| 24 | scheduleServer.destory(); |

|  |  |
| --- | --- |
| 25 | scheduleServer.destroySystemObjects(); |
| 26 | scheduleServer.destoryListeners(); |

|  |  |
| --- | --- |
| 27 | scheduleServer.destoryContext(); |
| 28 | scheduleServer.shutdown(); |

|  |  |
| --- | --- |
| 29 | } |
| 30 |  |

|  |  |
| --- | --- |
| 31 | } |

The above class ScheduleServerFacade is the facade class, which wraps a ScheduleServer object, it instantiates the server object through its constructor, and has two simple methods: startServer() and stopServer(). These methods internally perform the starting and the stopping of the server. The client is just needs to call these simple methods. Now, there is no need to call all the lifecycle and destroy methods, just the simple methods and the rest of the process will be performed by the facade class.

The code below shows how facade makes a complex interface simple to use.

View source prints

|  |  |
| --- | --- |
| 01 | package com.javacodegeeks.patterns.facadepattern; |
| 02 |  |

|  |  |
| --- | --- |
| 03 | public class TestFacade { |
| 04 |  |

|  |  |
| --- | --- |
| 05 | public static void main(String[] args) { |
| 06 |  |

|  |  |
| --- | --- |
| 07 | ScheduleServer scheduleServer = new ScheduleServer(); |
| 08 | ScheduleServerFacade facadeServer = new ScheduleServerFacade(scheduleServer); |

|  |  |
| --- | --- |
| 09 | facadeServer.startServer(); |
| 10 |  |

|  |  |
| --- | --- |
| 11 | System.out.println("Start working......"); |
| 12 | System.out.println("After work done........."); |

|  |  |
| --- | --- |
| 13 |  |
| 14 | facadeServer.stopServer(); |

|  |  |
| --- | --- |
| 15 | } |
| 16 |  |

|  |  |
| --- | --- |
| 17 | } |

Also, please note that, although the facade class has provided a simple interface to the complex subsystem, it has not encapsulated the subsystem. A client can still access the low-level interfaces of the subsystem. So, a facade provides an extra layer, a simple interface to the complex subsystem, but it does not completely hide the direct accessibility to the low-level interfaces of the complex subsystem.

3. Use of the Facade Pattern

Use the Facade Pattern, when:

1. You want to provide a simple interface to a complex subsystem. Subsystems often get more complex as they evolve. Most patterns, when applied, result in more and smaller classes. This makes the subsystem more reusable and easier to customize, but it also becomes harder to use for clients that don’t need to customize it. A facade can provide a simple default view of the subsystem that is good enough for most clients. Only clients needing more customizability will need to look beyond the facade.
2. There are many dependencies between clients and the implementation classes of an abstraction. Introduce a facade to decouple the subsystem from clients and other subsystems, thereby promoting subsystem independence and portability.
3. You can layer your subsystems. Use a facade to define an entry point to each subsystem level. If subsystems are dependent, then you can simplify the dependencies between them by making them communicate with each other solely through their facades.

In coming days I will be sharing many more materials on different design patterns………….!!!

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, September 22, 2015 1:14 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Rajnish K Tiwari; Sandeep Palo; Radhika Danda  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 9( Puzzles)-Very imp

Adding Sandeep and Radhika Danda .

.

This highly confidential for GS  Account and GS academy . Please do n’ t circulate to any or do n’ t open outside IGATE premises

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, September 14, 2015 5:10 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Rajnish K Tiwari  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 9( Puzzles)-Very imp   
**Importance:** High

Dear All ,

The greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time.

**Thomas Alva Edison inventor** [light bulb](https://en.wikipedia.org/wiki/Incandescent_light_bulb)

Please try to solve this by tomorrow 1st half(Please ignore who has cleared the GS interview )

**Puzzle** 1.You have a birthday cake and have exactly 3 cuts to cut it into 8 equal pieces. How do you do it?

**Puzzle** 2. Given the numbers 1 to 1000, what is the minimum number of guesses needed to find a specific number if you are given the hint ‘higher’ or ‘lower’ for each guess you make?

**Puzzle** 3. Suppose that you are standing in a hallway next to 3 light switches, which are all off. There is another room down the hallway, where there are 3 incandescent light bulbs – each light bulb is operated by one of the switches in the hallway. Because the light bulbs are in another room, you can not see them since you are standing in the hallway. How would you figure out which switch operates which light bulb, if you can only go the room with the light bulbs one time, and only one time?

**Puzzle** 4. You are standing in a school hallway lined with 100 closed lockers. You then open all 100 lockers. After this, you then close every 2nd locker (so the 2nd, 4th, 6th…98th and 100th are all closed). Then, you go to every third locker and open it if it is closed or close it if it is open (let’s call this toggling the locker for our discussion). You proceed to toggle every nth locker on pass number n. So, for example, on pass number 16 – you will toggle every 16th locker. After your hundredth pass of the hallway, in which you toggle only locker number 100, how many lockers are now open?In a hall with x lockers, how many lockers remain open after pass number x?

**Puzzle** 5. You are given 8 pennies, 7 of which weigh exactly the same, but one penny weighs less than the other 7. You also have a judge scale. Find the one penny that weighs the least in *less* than 3 steps.

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Friday, September 11, 2015 4:53 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav; Rajnish K Tiwari  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 8( JVM and Garbage Collector Tunning)-Very imp   
**Importance:** High

Dear  All ,

Please make sure you all have completed all the assignment that we have given on behalf of GS Academy in last one week. I have got very few responses till now.

It is suggested to revise all the new adv. concept regularly as human brain start erasing thing from the memory after a week’s time if you not revise it properly. So it really imp to increase your knowledge base and revise everything on regular interval of

time.

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Thursday, September 10, 2015 12:40 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav; Rajnish K Tiwari  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 8( JVM and Garbage Collector Tunning)-Very imp   
**Importance:** High

Dear All ,

Please prepare JVM  and Garbage Collector tuning very thoroughly they are asking lots of question now a days on this .

# Java Performance Tuning, Profiling, and Memory Management

Java application performance is an abstract word until you face its real implications. It may vary depending on your interpretation of the word 'performance'. This article is meant to give the developer a perspective of the various aspects of the JVM internals, the controls and switches that can be altered to optimal effects that suit your application. There is no single size that can fits all. You need to customize to suit your application.

You may be facing one of the issues listed below:

1. The dreaded java.lang.OutOfMemory Error
2. Your application is literally crawling.

Before we take the plunge into solving the issues, we first need to understand some of the theory behind the issues.

### ****Theory****

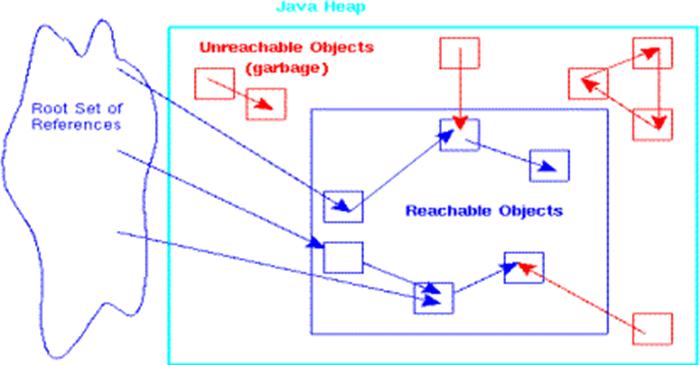
What does the JVM do? The Java Virtual Machine has two primary jobs:

1. Executes Code
2. Manages Memory  
   This includes allocating memory from  the OS, managing Java allocation including heap compaction,  
   and removal of garbaged objects

Besides the above, the JVM also does stuff like managing monitors.

### Very Basic Java Theory

An object is created in the heap and is garbage-collected after there are no more references to it. Objects cannot be reclaimed or freed by explicit language directives. Objects become garbage when they’re no longer reachable from the root set (e.g static objects)



Objects inside the blue square are reachable from the thread root set, while objects outside the square (in red) are not.

The sequence of the garbage collection process is as follows:

1. Root set tracing and figure out objects that are not referenced at all.

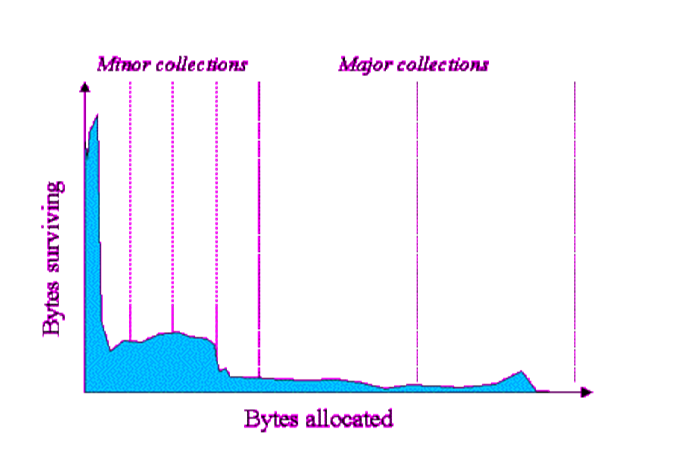
2. Put the garbage objects from above in finalizer Q

3. Run finalize() of each of these instances

4. Free memory

### ****Infant mortality in Java****

Most of the objects (80%) in a typical Java application die young. But this may not be  true for your application. Hence there is a need to figure out this rough infant mortality number so that you can tune the JVM accordingly.



### ****JVM flavors****

The Sun JVM understands the options -classic, -client and -server

* classic : disables the Hotspot JIT compiler.
* client (default): activates the Hotspot JIT for "client" applications.
* server: activates the "server" Hotspot JIT: it requires a fair amount of time to warm up, but delivers best performance for server.

Don't forget that, if you use them, -server or -client must be the first argument to Java.  
  
The Hotspot JVM uses adaptive optimization

* JVM begins by interpreting all code, but it monitors the HotSpot
* Fires off a background thread that compiles hotspot bytecode to native code
* Hotspot JVM is only compiling and optimizing the "hot spot". Hotspot JVM has more time than a traditional JIT to perform optimizations
* The Hotspot JVM keeps the old bytecodes around in case a method moves out of the hot spot.

### ****Java Garbage Collector****

The following describes what the Java Garbage Collector does.

**Sun Classic (1.1 JVM) ...for historical reasons**

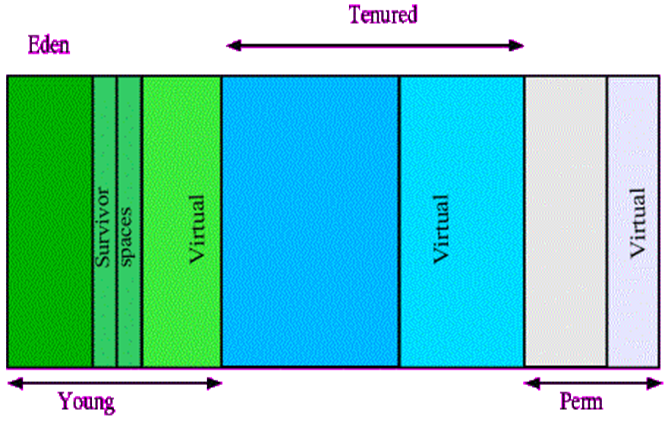
* Mark, Sweep & Compact  
    Mark: identify garbage  
    Sweep: Find garbage on heap, de-allocate it  
    Compact: collect all empty memory together
* Eligibility for garbage collection is determined by walking across memory, determining reachability and then compacting the heap
* Compaction is just copying the live objects so that they’re adjacent in memory
* there’s one large, contiguous block of free memory
* The main problem with classic mark, sweep and compact is that all other threads have to be suspended while the garbage collector runs
* Pause time is proportional to the number of objects on the heap

**Sun HotSpot( 1.2+ JVM)**

* Sun improved memory management in the Java 2 VMs by switching to a generational garbage collection scheme.
* The JavaHeap is separated into two regions(we will exclude the Permanent Generation for the time being):  
   New Objects  
   Old Objects
* The New Object Regions is subdivided into three smaller regions:  
   1. **Eden**, where objects are allocated  
   2. Survivor semi-spaces:  **From**and **To**
* The Eden area is set up like a stack - an object allocation is implemented as a pointer increment. When the Eden area is full, the GC does a reachability test and then copies all the live objects from Eden to the To region.
* The labels on the regions are swapped
* To becomes  From - now the  From area has objects.

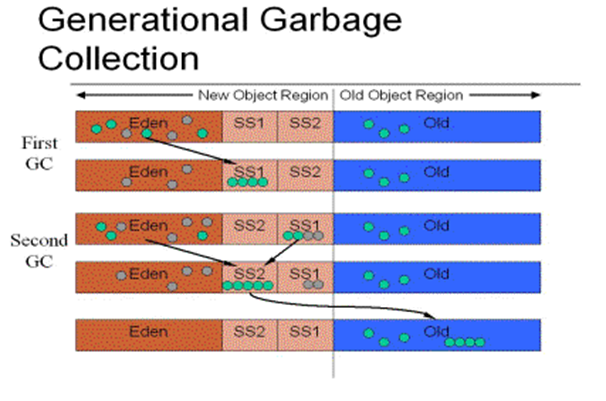
### ****JVM Heap****:

Java Heap is divided into 3 generations: Young(Eden), Old(Tenured), and Permanent.



**Arrangement of generations:**

The diagram below shows how objects get created in New generation and then move to survivor Spaces at every GC run, and if they survive for long to be considered old, they get moved to the Tenured generation. The number of times an object need to survive GC cycles to be considered old enough can be configured.



By default, Java has 2 separate  threads for GC, one each for young(minor GC) and old generation(major GC). The minor GC (smaller pause, but more frequent) occurs to clean up garbage in the young generation, while  the major GC (larger pause, but less frequent) cleans up the  garbage  in the old  generation. If the major GC too fails to free required memory, the JVM increases the current memory to help create new object. This whole  cycle  can go on till the current memory reaches the MaxMemory for the JVM (default is 64MB for client JVM), after which JVM throws OutOfMemory Error.

### Permanent Generation

Class information is stored in the perm generation. Also constant strings are stored there. Strings created dynamically in your application with String.intern() will also be stored in the perm generation.Reflective objects (classes, methods, etc.) are stored in perm.It holds all of the reflective data for the JVM

**JVM process memory**

The windows task manager just shows the memory usage of the java.exe task/process. It is not unusual for the total memory consumption of the VM to exceed the value of -Xmx

**Managed Heap**(java heap, PERM, code cache) + **NativeHEAP** +  **ThreadMemory** <= 2GB (PAS on windows)

Code-cache contains JIT code and hotspot code.  
       ThreadMemory = Thread\_stack\_size\*Num\_threads.

ManagedHeap: Managed by the developer.

Java heap: This part of the memory is used when you create new java objects.

Perm: for relfective calls etc.

NativeHeap : Used for native allocations.

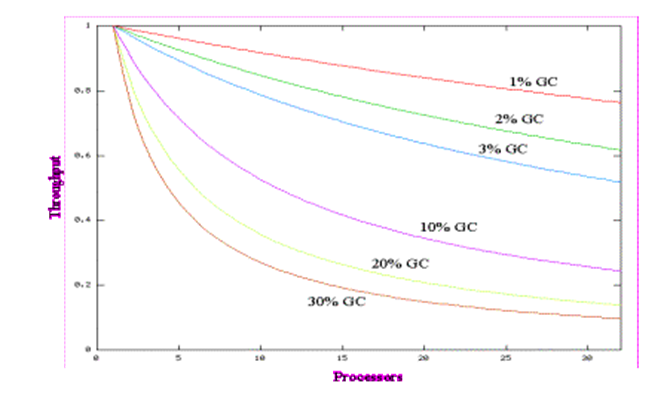
ThreadMemory: used for thread allocations.

What you see  in the TaskManager is the total PAS, while what the profiler shows is the Java Heap and the PERM(optionally)

**Platforms                       Maximum PAS\***

1. x86 / Redhat Linux 32 bit         2 GB
2. x86 / Redhat Linux 64 bit         3 GB
3. x86 / Win98/2000/NT/Me/XP    2 GB
4. x86 / Solaris x86 (32 bit)         4 GB
5. Sparc / Solaris 32 bit               4 GB

### Why GC needs tuning



**Limits of Vertical scaling**

If F is the fraction of a calculation that is sequential (i.e. cannot benefit from parallelization), and (1 − F) is the fraction that can be parallelized, then the maximum speedup that can be achieved by using N processors is:  
  
     1  
------------    **Amdahl's law**  
F + (1-F)/N

In the limit, as N -> infinity, the maximum speedup tends to 1/F. If F is only 10%, the problem can be sped up by only a maximum of a factor of 10, no matter how large the value of N used.

So we assume that there is a scope of leveraging benefits of multiple CPUs or multithreading.All right, enough of theory..........can it solve my problem??

### ****Problem Statements****

1. **Application slow**  
     
   Your application may be crawling because it's spending too much time cleaning up the garbage , rather than running the app.  
     
   Solution: Need to tune the JVM parameters. Take steps to Balance b/w pause and GC freq.
2. **Consumes too much memory**  
   The memory footprint  of  the  application is  related  to the number and  size of the live objects that are in the JVM at  any  given point of time. This  can be either due to valid objects that  are  required to stay in memory, or because programmer forgot to remove the reference to unwanted objects (typically known as 'Memory leaks' in java parlance. And as  the memory footprint hits the threshold, the JVM throws the java.lang.OutOfMemoryError.

**Java.lang.OutOfMemoryError can occur due to 3 possible reasons:**  
  
1. JavaHeap space low to create new objects . Increase by -Xmx    (java.lang.OutOfMemoryError: Java heap space).  
java.lang.OutOfMemoryError:**Java heap space**  
MaxHeap=30528 KB  TotalHeap=30528 KB FreHeap=170 KB  UsedHeap=**30357**KB  
  
2. Permanent Generation low. Increase by XX:MaxPermSize=256m (java.lang.OutOfMemoryError: PermGen space)  
java.lang.OutOfMemoryError:**PermGen space**  
MaxHeap=65088 KB  TotalHeap=17616 KB      FreeHeap=9692 KB  UsedHeap=**7923**KB

Heap

def new generationtotal 1280K, used 0K [0x02a70000, 0x02bd0000, 0x02f50000)

eden space 1152K,0% used [0x02a70000, 0x02a70000, 0x02b90000)

from space 128K,0% used [0x02bb0000, 0x02bb0000, 0x02bd0000)

tospace 128K,0% used [0x02b90000, 0x02b90000, 0x02bb0000)

tenured generationtotal 16336K, used 7784K [0x02f50000, 0x03f44000, 0x06a70000)

the space 16336K,47% used [0x02f50000, 0x036ea3f8, 0x036ea400, 0x03f44000)

compacting perm gen**total 12288K, used 12287K** [0x06a70000, 0x07670000, 0x07670000)

the space 12288K,**99% used**[0x06a70000, 0x0766ffd8, 0x07670000, 0x07670000)

3. java.lang.OutOfMemoryError:**....**Out of swap space ...

JNI Heap runs low on memory, even though the JavaHeap and the PermGen have memory. This typically happens if you are  meking lots of heavy JNI calls, but the JavaHeap objects occupy little space. In that scenario the GC might not feel the urge to cleanup JavaHeap, while the JNI Heap keeps on increasing till it goes out of memory.

If you use java NIO packages, watch out for this issue. DirectBuffer allocation uses the native heap.

The NativeHeap can be increasded by -XX:MaxDirectMemorySize=256M (default is 128)

### ****Diagnosis:****

There are some starting points to diagnose the problem.You may start with the '-verbose:gc' flag on the java command and see the memory footprint as the application progresses, till you find a spike. You may analyze the logs or use a light profiler like JConsole (part of JDK) to check the memory graph. If you need the details of the objects that are occupying the memory at a certain point, then you may use JProfiler or AppPerfect which can provide the details of each object instance and all the in/out bound  references to/from it. This is a memory intensive procedure and not meant for production systems. Depending upon your application, these heavy profilers can slow down the app upto 10 times.

Below are  some of the ways you can zero-in on the issue.

**A) GC outputs**  
-verbose:gc

This flag starts printing additional lines to the console, like given below

[GC 65620K -> 50747K(138432K), 0.0279446 secs]  
[Full GC 46577K -> 18794K(126848K), 0.2040139 secs]    
Combined size of live objects before(young+tenured) GC -> Combined size of live objects(young+tenured)  after GC (Total heap size, not counting the space in the permanent generation  
-XX:+PrintHeapAtGC : More details  
•-XX:+PrintGCTimeStamps will additionally print a time stamp at the start of each collection.  
111.042: [GC 111.042: [DefNew: 8128K->8128K(8128K), 0.0000505 secs]  
111.042: [Tenured: 18154K->2311K(24576K), 0.1290354 secs]  
26282K->2311K(32704K), 0.1293306 secs]  
The collection starts about 111 seconds into the execution of the application. The tenured generation usage was reduced to about 10%  
18154K->2311K(24576K)  
  
**B) hprof output file**  
java –Xrunhprof:heap=sites,cpu=samples,depth=10,thread=y,doe=y  
The heap=sites tells the profiler to write information about memory utilization on the heap, indicating where it was allocated.  
cpu=samples tells the profiler to do statistical sampling to determine CPU use.  
depth=10 indicates the depth of the trace for threads.  
thread=y tells the profiler to identify the threads in the stack traces.  
doe=y tells the profiler to produce dump of profiling data on exit.

**C) -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=C:\OOM.txt**

Dump the heap on OOM, and then analyze the OOM.txt (Binary file) with **jhat**tool (bundled with JDK)

The command below will launch http server  @port 7777 . Open a browser with the URL 'http://localhost:7777'  to see the results.

jhat -port 7777 c:\OOM.txt

**D) Profiling the app**

You can profile the application to figure out Memory Leaks.

Java memory leaks(or what we like to call unintentionally retained objects), are often caused by saving an object reference in a class level collection and forgetting to remove it at the proper time. The collection might be storing 100 objects, out of which 95 might never be used. So in this case those 95 objects are creating the memory leak, since the GC cannot free them as they are referenced by the collection.

There are also other kinds of problems with managing resources that impact performance, such as not closing JDBC Statements/ResultSets in a finally block (many JDBC drivers store a Statement reference in the Connection object).

A java "memory leak" is more like holding a strong reference to an object though it would never be needed anymore. The fact that you hold a strong reference to an object prevents the GC from deallocating it.. Java "memory leaks" are objects that fall into category (2). Objects that are reachable but not "live" can be considered memory leaks.

JVMPI for Profiling applications  give a high level of detailing  
Profilers: Hprof, **JConsole**, **JProfiler**, **AppPerfect**, YourKit, Eclipse Profiler, NetBeans Profiler ,JMP, Extensible Java Profiler (EJP), TomcatProbe, Profiler4j

JConsole is good for summary level info, tracking the memory footprint, checking Thread deadlocks etc. It does not provide details of the Heap object. For Heap details you may use AppPerfect (licensed) or JProfiler.

**E) For NativeHeap issues.....**

JRockit JDK (from BEA) provides better tools than the SUN JDK to peep inside the JNI Heap(atleast on Windows).

JRockt Runtime  Analyzer ...this  is part of the jrockit  install.  
jrcmd PSID print\_memusage  
JRMC.exe  ...launch from /bin and  start  recording.

### ****Try to get some Solution****:

Based on the findings from the diagnosis, you may have to take these actions:

1. **Code change** - For memory leak issues, it has to be a code change.
2. **JVM parameters tuning** - You need to find the behavior of your app in terms of the ratio of young to old objects, and then tune the JVM accordingly. We ll talk abt when to tune a parameter as we discuss the relevant params below.  
     
   Memory parameters:  
     
   **Memory Size:**  overall size, individual region sizes  
     
   -ms, -Xms     
   sets the initial heap size (young and tenured generation ONLY, NOT Permanent)  
     
   If the app starts with a large memory footprint, then you should set the initial heap to a large value so that the JVM does not consume cycles to keep expanding the heap.  
     
   -mx, -Xmx  
   sets the maximum heap size(young and tenured gen ONLY,NOT Perm) (default:  64mb)  
     
   This is the most frequently tuned parameter to suit the max memory requirements of the app. A low value  overworks the GC so that it frees space for new objects to be  created, and may lead to OOM. A very high value can starve other apps and induce swapping. Hence, Profile the memory requirements to select the right value.  
     
   -XX:PermSize=256 -XX:MaxPermSize=256m  
     
   MaxPermSize  default value (32mb for -client and 64mb for -server)  
   Tune this to increase the Permanent gereration max size.
3. **GC parameters:**  
   -X**minf** [0-1], -XX:**MinHeapFreeRatio**[0-100]  
     
   sets the percentage of minimum free heap space - controls heap expansion rate  
     
   -X**maxf** [0-1], -XX:**MaxHeapFreeRatio**[0-100]  
   sets the percentage of maximum free heap space - controls when the VM will return unused heap memory to the OS  
   -XX:**NewRatio**  
   sets the ratio of the old and new generations in the heap. A NewRatio of 5 sets the ratio of new to old at 1:5, making the new generation occupy 1/6th of the overall heap  
   defaults: client 8, server 2  
   -XX:**SurvivorRatio**  
   sets the ratio of the survivor space to the eden in the new object area. A SurvivorRatio of 6 sets the ratio of the three spaces to 1:1:6, making each survivor space 1/8th of the new object region

### ****Garbage Collector Tuning:****

Types of GarbageCollectors (not complete list)

1. Throughput collector: (default for Server JVM)  
   •parallel version of the young generation collector.  
   •-XX:+UseParallelGC  
   •The tenured gc is the same as the serial collector (default GC for client JVM).  
   •multiple threads to execute a minor collection  
   •application has a large number of threads allocating objects / large Eden  
   •-XX:+UseParallelOldGC (major also in parallel)
2. Concurrent low pause collector  :  
   •collects the tenured generation and does most of the collection concurrently with the execution of the application. Attempts to reduce the pause times needed to collect the tenured generation  
   •-Xincgc™ or -XX:+UseConcMarkSweepGC  
   •The application is paused for short periods during the collection. A parallel version of the young generation copying collector is used with the concurrent collector.  
   •Multiprocessor; apps that have a relatively large set of long-lived data (a large tenured generation;  
   •Apps where response time is more important than overall throughput e.g. JAVA\_OPTS= -Xms128M -Xmx1024M -XX:NewRatio=1 -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -Xloggc:E:\loggc.txt  
   FlipSide: Synchronization overhead, Fragmentation

**Performance Solution**

1. Application Software profiling
2. Server and JVM tuning
3. Right Hardware and OS
4. Code improvement as per the Behaviour of your application & profiling results….…. easier said than done
5. Use JVM the right way : optimal JVM params
6. Client / server application
7. •-XX:+UseParallelGC  if u have multiprocessors

**Some Tips**

* Unless you have problems with pauses, try granting as much memory as possible to the virtual machine
* Setting -Xms and -Xmx to the same value ….but be sure about the application behaviour
* Be sure to increase the memory as you increase the number of processors, since allocation can be parallelized
* Don’t forget to tune the Perm generation
* Minimize the use of synchronization
* Use multithreading only if it benefits. Be aware of the thread overheads. e.g a simple task like counter incrementing from 1 to billion ....use single thread. Multiple threads will ruin to mutiple of 10. I tested it out on dual CPU WinXP with 8 threads.
* Avoid premature object creation. Creation should be as close to the actual place of use as possible. Very basic concept that we  tend to overlook.
* JSPs are generally slower than servlets.
* Too many custom CLs, reflection : increase Perm generation. Don't be PermGen-agnostic.
* Soft References for memory leakages. They enable smart caches and yet do not load memory. GC will flush out SoftReferences automatically if the JVM runs low on memory.
* StringBuffer instead of String concat
* Minimize JNI calls in your code
* XML APIs – be careful …SAX or DOM- make correct choice. Use precompiled xpaths for better performance of the queries.

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Thursday, September 10, 2015 12:17 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 6  
**Importance:** High

# JVM(Java) Heap size

Please go thu it concisely . In the recent  past they have asked lots of question on the **Java Heap**

Two JVM options are often used to tune JVM heap size: -Xmx for maximum heap size, and -Xms for initial heap size. Here are some common mistakes made by developers while using them:

1. Missing m, M, g or G at the end (they are case insensitive). For example,

java -Xmx128 BigApp

java.lang.OutOfMemoryError: Java heap space

The correct command should be: java -Xmx128m BigApp. To be precise, -Xmx128 is a valid setting for very small apps, like HelloWorld. But in real life, I guess you really mean -Xmx128m

1. Extra space in JVM options, or incorrectly use =. For example,

java -Xmx 128m BigApp

Invalid maximum heap size: -Xmx

Could not create the Java virtual machine.

java -Xmx=512m HelloWorld

Invalid maximum heap size: -Xmx=512m

Could not create the Java virtual machine.

The correct command should be java -Xmx128m BigApp, with no whitespace nor =. -X options are different than -Dkey=value system properties, where = is used.

1. Only setting -Xms JVM option and its value is greater than the default maximum heap size, which is 64m. The default minimum heap size seems to be 0. For example,

java -Xms128m BigApp

Error occurred during initialization of VM

Incompatible initial and maximum heap sizes specified

The correct command should be java -Xms128m -Xmx128m BigApp. Its a good idea to set the minimum and maximum heap size to the same value. In any case, dont let the minimum heap size exceed the maximum heap size.

1. Heap size is larger than your computers physical memory. For example,

java -Xmx2g BigApp

Error occurred during initialization of VM

Could not reserve enough space for object heap

Could not create the Java virtual machine.

The fix is to make it lower than the physical memory: java -Xmx1g BigApp

1. Incorrectly use mb as the unit, where m or M should be used instead.

java -Xms256mb -Xmx256mb BigApp

Invalid initial heap size: -Xms256mb

Could not create the Java virtual machine.

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, September 09, 2015 4:34 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 6  
**Importance:** High

Dear All ,

Logic will get you from A to B. Imagination will take you everywhere.

Albert Einstein

Please try to solve the below mentioned five problem statement by Friday EOD:

* Given a list of words, find the absolute minimum distance between two words.
* Given a matrix A, write a matrix M for which every element [i,j] is the sum of all elements of A left and above A[i,j].
* Find the duplicate member in a sequential integer array starting with values starting from 1.
* Given n stacks of ints, find the maximum sum of k numbers by popping out any stacks.
* Find expected number of tosses of an unbiased or biased coin, until there are m (>= 1) heads in a row.

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Tuesday, September 08, 2015 2:50 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; Prakruti Mulimani; Sankar Theerthagiri; Bapu S Jadhav; KARTHICK SANTHARAM; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 5  
**Importance:** High

Dear All ,

[Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop learning.](http://www.brainyquote.com/quotes/quotes/a/alberteins125368.html)

[**Albert Einstein**](http://www.brainyquote.com/quotes/quotes/a/alberteins125368.html)

Please try to solve the below mentioned five Java coding puzzles by Thursday EOD:

|  |
| --- |
| **Java Coding Puzzle 1**  Try compiling and running the code below - then uncomment for loop compile and run.  Why does this program have an error when for loop is commented out?  **public class** JavaMemoryPuzzlePolite {  **private final** **int** dataSize = (**int**)(Runtime.*getRuntime*().maxMemory()\* 0.6);  **public** **void** f(){  {  System.*out*.println(dataSize);  **byte**[] data = **new** **byte**[dataSize];  }  /\*for(int i = 0; i < 10; i++){  System.out.println("Please be so kind and release memory");  }\*/  System.*out*.println(dataSize);  **byte**[] data2 = **new** **byte**[dataSize];  }  **public** **static** **void** main(String []args){  JavaMemoryPuzzlePolite jmp = **new** JavaMemoryPuzzlePolite();  jmp.f();  }  } |
| **Java Coding Puzzle 2**  **Java Puzzle: 2 threads - 1 variable** Given a program:   public class ThreadsLimits extends Thread {  static volatile int x;   public void run() {  for (int i = 0; i < 10; i++){  int temp = x;  temp++;  x = temp;  }  }   public static void main(String[] args) {  Thread t1 = new ThreadsLimits();  Thread t2 = new ThreadsLimits();   t1.start();  t2.start();   }   }   What can you say about the value of x by the end of the program? (Assume both threads ended its execution) |
| **Java Coding Puzzle 3**  What does the following print (assuming it compiles, which it does):  public class StrungOut {  public static void main(String[] args) {  String s = new String("Hello world");  System.out.println(s);  }  }   class String{  private final java.lang.String s;  public String(java.lang.String s){  this.s = s;  }  public java.lang.String toString(){  return s;  }  } |
| **Java Coding Puzzle 4**  Ok here is another entertaining one: If you run this program 1000 times, what will it print most(if not all) of the times?  import java.util.Random;  public class Rhymes {  private static Random rnd = new Random();  public static void main(String[] args) { StringBuffer word = null;  switch(rnd.nextInt(2)){ case 1: word = new StringBuffer('P'); case 2: word = new StringBuffer('G'); default: word = new StringBuffer('M'); } word.append('a'); word.append('i'); word.append('n'); System.out.println(word); }  } |
| **Java Coding Puzzle 5** |
| **What does this program print?**  /\* MIND THE GAP \*/  import java.io.\*;  public class Gap {  private static final int GAP\_SIZE = 10 \* 1024;  public static void main(String[] args) throws Exception{  File tmp = File.createTempFile("gap", ".txt");  FileOutputStream out = new FileOutputStream(tmp);  out.write(1);  out.write(new byte[GAP\_SIZE]);  out.write(2);  out.close();  InputStream in = new BufferedInputStream(new FileInputStream(tmp));  int first = in.read();  in.skip(GAP\_SIZE);  int last = in.read();  System.out.println(first \* last);  }   } |

Please  try all these on Eclipse IDE without fail .

Please circulate this mail if I have missed anyone in the list.

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, September 07, 2015 5:52 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; Prakruti Mulimani; Sankar Theerthagiri; Bapu S Jadhav; KARTHICK SANTHARAM; Ravikiran Govindareddy; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 4  
**Importance:** High

## Sorting

## Please try to solve these problems

|  |
| --- |
| **Question 1** |

What is recurrence for worst case of QuickSort and what is the time complexity in Worst case?

|  |  |
| --- | --- |
| A | Recurrence is T(n) = T(n-2) + O(n) and time complexity is O(n^2) |
| B | Recurrence is T(n) = T(n-1) + O(n) and time complexity is O(n^2) |
| C | Recurrence is T(n) = 2T(n/2) + O(n) and time complexity is O(nLogn) |
| D | Recurrence is T(n) = T(n/10) + T(9n/10) + O(n) and time complexity is O(nLogn) |

|  |
| --- |
| **Question 2** |

Suppose we have a O(n) time algorithm that finds median of an unsorted array. Now consider a QuickSort implementation where we first find median using the above algorithm, then use median as pivot. What will be the worst case time complexity of this modified QuickSort.

|  |  |
| --- | --- |
| A | O(n^2 Logn) |
| B | O(n^2) |
| C | O(n Logn Logn) |
| D | O(nLogn) |

|  |
| --- |
| **Question 3** |

Which of the following is not a stable sorting algorithm in its typical implementation.

|  |  |
| --- | --- |
| A | Insertion Sort |
| B | Merge Sort |
| C | Quick Sort |
| D | Bubble Sort |

|  |
| --- |
| **Question 4** |

Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced).

|  |  |
| --- | --- |
| A | Quick Sort |
| B | Heap Sort |
| C | Merge Sort |
| D | Insertion Sort |

|  |
| --- |
| **Question 5** |

Given an unsorted array. The array has this property that every element in array is at most k distance from its position in sorted array where k is a positive integer smaller than size of array. Which sorting algorithm can be easily modified for sorting this array and what is the obtainable time complexity?

|  |  |
| --- | --- |
| A | Insertion Sort with time complexity O(kn) |
| B | Heap Sort with time complexity O(nLogk) |
| C | Quick Sort with time complexity O(kLogk) |
| D | Merge Sort with time complexity O(kLogk) |

|  |
| --- |
| **Question 6** |

Consider a situation where swap operation is very costly. Which of the following sorting algorithms should be preferred so that the number of swap operations are minimized in general?

|  |  |
| --- | --- |
| A | Heap Sort |
| B | Selection Sort |
| C | Insertion Sort |
| D | Merge Sort |

|  |
| --- |
| **Question 7** |

Which of the following is not true about comparison based sorting algorithms?

|  |  |
| --- | --- |
| A | The minimum possible time complexity of a comparison based sorting algorithm is O(nLogn) for a random input array |
| B | Any comparison based sorting algorithm can be made stable by using position as a criteria when two elements are compared |
| C | Counting Sort is not a comparison based sorting algortihm |
| D | Heap Sort is not a comparison based sorting algorithm. |

|  |
| --- |
| **Question 8** |

Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this: 2 5 1 7 9 12 11 10 Which statement is correct?

|  |  |
| --- | --- |
| A | The pivot could be either the 7 or the 9. |
| B | The pivot could be the 7, but it is not the 9 |
| C | The pivot is not the 7, but it could be the 9 |
| D | Neither the 7 nor the 9 is the pivot. |

|  |
| --- |
| **Question 9** |

Suppose we are sorting an array of eight integers using heapsort, and we have just finished some heapify (either maxheapify or minheapify) operations. The array now looks like this: 16 14 15 10 12 27 28 How many heapify operations have been performed on root of heap?

|  |  |
| --- | --- |
| A | 1 |
| B | 2 |
| C | 3 or 4 |
| D | 5 or 6 |

|  |
| --- |
| **Question 10** |

What is the best time complexity of bubble sort?

|  |  |
| --- | --- |
| A | N^2 |
| B | NlogN |
| C | N |
| D | N(logN)^2 |

|  |
| --- |
| **Question 11** |

You have to sort 1 GB of data with only 100 MB of available main memory. Which sorting technique will be most appropriate?

|  |  |
| --- | --- |
| A | Heap sort |
| B | Merge sort |
| C | Quick sort |
| D | Insertion sort |

|  |
| --- |
| **Question 12** |

What is the worst case time complexity of insertion sort where position of the data to be inserted is calculated using binary search?

|  |  |
| --- | --- |
| A | N |
| B | NlogN |
| C | N^2 |
| D | N(logN)^2 |

|  |
| --- |
| **Question 13** |

The tightest lower bound on the number of comparisons, in the worst case, for comparison-based sorting is of the order of

|  |  |
| --- | --- |
| A | N |
| B | N^2 |
| C | NlogN |
| D | N(logN)^2 |

|  |
| --- |
| **Question 14** |

In a modified merge sort, the input array is splitted at a position one-third of the length(N) of the array. What is the worst case time complexity of this merge sort?

|  |  |
| --- | --- |
| A | N(logN base 3) |
| B | N(logN base 2/3) |
| C | N(logN base 1/3) |
| D | N(logN base 3/2) |

|  |
| --- |
| **Question 15** |

Which sorting algorithm will take least time when all elements of input array are identical? Consider typical implementations of sorting algorithms.

|  |  |
| --- | --- |
| A | Insertion Sort |
| B | Heap Sort |
| C | Merge Sort |
| D | Selection Sort |

|  |
| --- |
| **Question 16** |

A list of n string, each of length n, is sorted into lexicographic order using the merge-sort algorithm. The worst case running time of this computation is (A) Description: O (n log n  (B) Description: O (n^2 log n)  (C) Description: O (n^2 + log n)  (D) Description: O (n^2) 

|  |  |
| --- | --- |
| A | A |
| B | B |
| C | C |
| D | D |

|  |
| --- |
| **Question 17** |

In quick sort, for sorting n elements, the (n/4)th smallest element is selected as pivot using an O(n) time algorithm. What is the worst case time complexity of the quick sort? (A) Description: \theta(n) (B) Description: \theta(nLogn) (C) Description: \theta(n^2) (D) Description: \theta(n^2 log n)

|  |  |
| --- | --- |
| A | A |
| B | B |
| C | C |
| D | D |

|  |
| --- |
| **Question 18** |

Consider the Quicksort algorithm. Suppose there is a procedure for finding a pivot element which splits the list into two sub-lists each of which contains at least one-fifth of the elements. Let T(n) be the number of comparisons required to sort n elements. Then

|  |  |
| --- | --- |
| A | T(n) <= 2T(n/5) + n |
| B | T(n) <= T(n/5) + T(4n/5) + n |
| C | T(n) <= 2T(4n/5) + n |
| D | T(n) <= 2T(n/2) + n |

|  |
| --- |
| **Question 19** |

Which of the following sorting algorithms has the lowest worst-case complexity?

|  |  |
| --- | --- |
| A | Merge Sort |
| B | Bubble Sort |
| C | Quick Sort |
| D | Selection Sort |

|  |
| --- |
| **Question 20** |

Which sorting algorithms is most efficient to sort string consisting of ASCII characters?

|  |  |
| --- | --- |
| A | Quick sort |
| B | Heap sort |
| C | Merge sort |
| D | Counting sort |

|  |
| --- |
| **Question 21** |

The number of elements that can be sorted in Description: \Theta(logn) time using heap sort is

(A) Description: \Theta(1)

(B) Description: \Theta(\sqrt{logn})

(C) Description: \Theta(Log n/(Log Log n))

(d) Description: \Theta(Log n)

|  |  |
| --- | --- |
| A | A |
| B | B |
| C | C |
| D | D |

|  |
| --- |
| **Question 22** |

Which of the following is true about merge sort?

|  |  |
| --- | --- |
| A | Merge Sort works better than quick sort if data is accessed from slow sequential memory. |
| B | Merge Sort is stable sort by nature |
| C | Merge sort outperforms heap sort in most of the practical situations. |
| D | All of the above. |

|  |
| --- |
| **Question 23** |

Given an array where numbers are in range from 1 to n6, which sorting algorithm can be used to sort these number in linear time?

|  |  |
| --- | --- |
| A | Not possible to sort in linear time |
| B | Radix Sort |
| C | Counting Sort |
| D | Quick Sort |

|  |
| --- |
| **Question 24** |

In quick sort, for sorting n elements, the (n/4)th smallest element is selected as pivot using an O(n) time algorithm. What is the worst case time complexity of the quick sort?<pre> (A) Description: \theta(n) (B) Description: \theta(nLogn) (C) Description: \theta(n^2) (D) Description: \theta(n^2 log n) </pre>

|  |  |
| --- | --- |
| A | A |
| B | B |
| C | C |
| D | D |

|  |
| --- |
| **Question 25** |

Consider the Quicksort algorithm. Suppose there is a procedure for finding a pivot element which splits the list into two sub-lists each of which contains at least one-fifth of the elements. Let T(n) be the number of comparisons required to sort n elements. Then

|  |  |
| --- | --- |
| A | T(n) <= 2T(n/5) + n |
| B | T(n) <= T(n/5) + T(4n/5) + n |
| C | T(n) <= 2T(4n/5) + n |
| D | T(n) <= 2T(n/2) + n |

|  |
| --- |
| **Question 26** |

Let P be a QuickSort Program to sort numbers in ascending order using the first element as pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2} respectively. Which one of the following holds?

|  |  |
| --- | --- |
| A | t1 = 5 |
| B | t1 < t2 |
| C | t1 > t2 |
| D | t1 = t2 |

|  |
| --- |
| **Question 27** |

You have an array of n elements. Suppose you implement [quicksort](http://geeksquiz.com/quick-sort/)by always choosing the central element of the array as the pivot. Then the tightest upper bound for the worst case performance is

|  |  |
| --- | --- |
| A | O(n2) |
| B | O(nLogn) |
| C | Theta(nLogn) |
| D | O(n3) |

|  |
| --- |
| **Question 28** |

In a permutation a1.....an of n distinct integers, an inversion is a pair (ai, aj) such that i < j and ai > aj. What would be the worst case time complexity of the [Insertion Sort](http://geeksquiz.com/insertion-sort/)algorithm, if the inputs are restricted to permutations of 1.....n with at most n inversions?

|  |  |
| --- | --- |
| A | Θ (n2) |
| B | Θ (n log n) |
| C | Θ (n1.5) |
| D | Θ (n) |

|  |
| --- |
| **Question 29** |

Randomized quicksort is an extension of quicksort where the pivot is chosen randomly. What is the worst case complexity of sorting n numbers using randomized quicksort?

|  |  |
| --- | --- |
| A | O(n) |
| B | O(n Log n) |
| C | O(n2) |
| D | O(n!) |

|  |
| --- |
| **Question 30** |

Which of the following changes to typical [QuickSort](http://geeksquiz.com/quick-sort/)improves its performance on average and are generally done in practice.

1) Randomly picking up to make worst case less

likely to occur.

2) Calling [insertion sort](http://geeksquiz.com/insertion-sort/) for small sized arrays

to reduce recursive calls.

3) QuickSort is [tail recursive](http://www.geeksforgeeks.org/tail-recursion/), so tail call

optimizations can be done.

4) A [linear time median searching algorithm](http://www.geeksforgeeks.org/kth-smallestlargest-element-unsorted-array-set-3-worst-case-linear-time/) is used

to pick the median, so that the worst case time

reduces to O(nLogn)

|  |  |
| --- | --- |
| A | 1 and 2 |
| B | 2, 3, and 4 |
| C | 1, 2 and 3 |
| D | 2, 3 and 4 |

|  |
| --- |
| **Question 31** |

Which one of the following is the recurrence equation for the worst case time complexity of the Quicksort algorithm for sorting n(≥ 2) numbers? In the recurrence equations given in the options below, c is a constant.

|  |  |
| --- | --- |
| A | T(n) = 2T (n/2) + cn |
| B | T(n) = T(n – 1) + T(0) + cn |
| C | T(n) = 2T (n – 2) + cn |
| D | T(n) = T(n/2) + cn |

|  |
| --- |
| **Question 32** |

Assume that a mergesort algorithm in the worst case takes 30 seconds for an input of size 64. Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 minutes?

|  |  |
| --- | --- |
| A | 256 |
| B | 512 |
| C | 1024 |
| D |  |

It is suggest to do it on eclipse ide and look for all the possible output. Try to find all the possible solutions for this with proper logic.

Please try to finish it by day after tomorrow eod.

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, September 07, 2015 5:37 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; Prakruti Mulimani; Sankar Theerthagiri; Bapu S Jadhav; KARTHICK SANTHARAM; Ravikiran Govindareddy; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik; Shrijith Shirnali  
**Subject:** RE: Logical and analytical assignment 3  
**Importance:** High

**Linked List**

|  |
| --- |
| **Question 1** |

What does the following function do for a given Linked List with first node as *head*?

|  |
| --- |
| void fun1(struct node\* head)  {  if(head == NULL)  return;  fun1(head->next);  printf("%d  ", head->data);  } |

|  |  |
| --- | --- |
| A | Prints all nodes of linked lists |
| B | Prints all nodes of linked list in reverse order |
| C | Prints alternate nodes of Linked List |
| D | Prints alternate nodes in reverse order |

|  |
| --- |
| **Question 2** |

Which of the following points is/are true about Linked List data structure when it is compared with array

|  |  |
| --- | --- |
| A | Arrays have better cache locality that can make them better in terms of performance. |
| B | It is easy to insert and delete elements in Linked List |
| C | Random access is not allowed in a typical implementation of Linked Lists |
| D | The size of array has to be pre-decided, linked lists can change their size any time. |
| E | All of the above |

|  |
| --- |
| **Question 3** |

Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as *prev*and next pointer as *next*.

|  |
| --- |
| void fun(struct node \*\*head\_ref)  {  struct node \*temp = NULL;  struct node \*current = \*head\_ref;  while (current !=  NULL)  {  temp = current->prev;  current->prev = current->next;  current->next = temp;  current = current->prev;  }  if(temp != NULL )  \*head\_ref = temp->prev;  } |

Assume that reference of head of following doubly linked list is passed to above function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

|  |  |
| --- | --- |
| A | 2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5 |
| B | 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6. |
| C | 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1. |
| D | 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2 |

|  |
| --- |
| **Question 4** |

Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity?

|  |  |
| --- | --- |
| A | Insertion Sort |
| B | Quick Sort |
| C | Heap Sort |
| D | Merge Sort |

|  |
| --- |
| **Question 5** |

The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

|  |
| --- |
| /\* Link list node \*/  struct node  {  int data;  struct node\* next;  };  /\* head\_ref is a double pointer which points to head (or start) pointer  of linked list \*/  static void reverse(struct node\*\* head\_ref)  {  struct node\* prev   = NULL;  struct node\* current = \*head\_ref;  struct node\* next;  while (current != NULL)  {  next  = current->next;  current->next = prev;  prev = current;  current = next;  }  /\*ADD A STATEMENT HERE\*/  } |

What should be added in place of "/\*ADD A STATEMENT HERE\*/", so that the function correctly reverses a linked list.

|  |  |
| --- | --- |
| A | \*head\_ref = prev; |
| B | \*head\_ref = current; |
| C | \*head\_ref = next; |
| D | \*head\_ref = NULL; |

|  |
| --- |
| **Question 6** |

What is the output of following function for start pointing to first node of following linked list? 1->2->3->4->5->6

|  |
| --- |
| void fun(struct node\* start)  {  if(start == NULL)  return;  printf("%d  ", start->data);  if(start->next != NULL )  fun(start->next->next);  printf("%d  ", start->data);  } |

Run on IDE

|  |  |
| --- | --- |
| A | 1 4 6 6 4 1 |
| B | 1 3 5 1 3 5 |
| C | 1 2 3 5 |
| D | 1 3 5 5 3 1 |

|  |
| --- |
| **Question 7** |

The following C function takes a simply-linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

|  |
| --- |
| typedef struct node  {  int value;  struct node \*next;  }Node;  Node \*move\_to\_front(Node \*head)  {  Node \*p, \*q;  if ((head == NULL: || (head->next == NULL))  return head;  q = NULL; p = head;  while (p-> next !=NULL)  {  q = p;  p = p->next;  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  return head;  } |

|  |  |
| --- | --- |
| A | q = NULL; p->next = head; head = p; |
| B | q->next = NULL; head = p; p->next = head; |
| C | head = p; p->next = q; q->next = NULL; |
| D | q->next = NULL; p->next = head; head = p; |

|  |
| --- |
| **Question 8** |

The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

|  |
| --- |
| struct node  {  int value;  struct node \*next;  };  void rearrange(struct node \*list)  {  struct node \*p, \* q;  int temp;  if ((!list) || !list->next)  return;  p = list;  q = list->next;  while(q)  {  temp = p->value;  p->value = q->value;  q->value = temp;  p = q->next;  q = p?p->next:0;  }  } |

Run on IDE

|  |  |
| --- | --- |
| A | 1,2,3,4,5,6,7 |
| B | 2,1,4,3,6,5,7 |
| C | 1,3,2,5,4,7,6 |
| D | 2,3,4,5,6,7,1 |

|  |
| --- |
| **Question 9** |

In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is (GATE CS 2002)

|  |  |
| --- | --- |
| A | log 2 n |
| B | n/2 |
| C | log 2 n – 1 |
| D | n |

|  |
| --- |
| **Question 10** |

Suppose each set is represented as a linked list with elements in arbitrary order. Which of the operations among union, intersection, membership, cardinality will be the slowest? (GATE CS 2004)

|  |  |
| --- | --- |
| A | union only |
| B | intersection, membership |
| C | membership, cardinality |
| D | union, intersection |

|  |
| --- |
| **Question 11** |

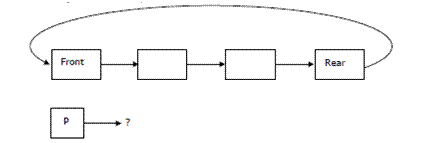
Consider the function f defined below.

|  |
| --- |
| struct item  {  int data;  struct item \* next;  };  int f(struct item \*p)  {  return (  (p == NULL) ||  (p->next == NULL) ||  (( P->data <= p->next->data) && f(p->next))  );  } |

For a given linked list p, the function f returns 1 if and only if (GATE CS 2003)

|  |  |
| --- | --- |
| A | the list is empty or has exactly one element |
| B | the elements in the list are sorted in non-decreasing order of data value |
| C | the elements in the list are sorted in non-increasing order of data value |
| D | not all elements in the list have the same data value. |

|  |
| --- |
| **Question 12** |

A circularly linked list is used to represent a Queue. A single variable p is used to access the Queue. To which node should p point such that both the operations enQueue and deQueue can be performed in constant time? (GATE 2004) [](http://d18khu5s3lkxd9.cloudfront.net/wp-content/uploads/2013/06/circularLinkedList.gif)

|  |  |
| --- | --- |
| A | rear node |
| B | front node |
| C | not possible with a single pointer |
| D | node next to front |

|  |
| --- |
| **Question 13** |

What are the time complexities of finding 8th element from beginning and 8th element from end in a singly linked list? Let n be the number of nodes in linked list, you may assume that n > 8.

|  |  |
| --- | --- |
| A | O(1) and O(n) |
| B | O(1) and O(1) |
| C | O(n) and O(1) |
| D | O(n) and O(n) |

|  |
| --- |
| **Question 14** |

Is it possible to create a doubly linked list using only one pointer with every node.

|  |  |
| --- | --- |
| A | Not Possible |
| B | Yes, possible by storing XOR of addresses of previous and next nodes. |
| C | Yes, possible by storing XOR of current node and next node |
| D | Yes, possible by storing XOR of current node and previous node |

|  |
| --- |
| **Question 15** |

Given pointer to a node X in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node X from given linked list?

|  |  |
| --- | --- |
| A | Possible if X is not last node. Use following two steps (a) Copy the data of next of X to X. (b) Delete next of X. |
| B | Possible if size of linked list is even. |
| C | Possible if size of linked list is odd |
| D | Possible if X is not first node. Use following two steps (a) Copy the data of next of X to X. (b) Delete next of X. |

|  |
| --- |
| **Question 16** |

You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list?

|  |  |
| --- | --- |
| A | Delete the first element |
| B | Insert a new element as a first element |
| C | Delete the last element of the list |
| D | Add a new element at the end of the list |

|  |
| --- |
| **Question 17** |

Consider the following function to traverse a linked list.

|  |
| --- |
| void traverse(struct Node \*head)  {  while (head->next != NULL)  {  printf("%d  ", head->data);  head = head->next;  }  } |

Which of the following is **FALSE**about above function?

|  |  |
| --- | --- |
| A | The function may crash when the linked list is empty |
| B | The function doesn't print the last node when the linked list is not empty |
| C | The function is implemented incorrectly because it changes head |

It is suggest to do it on eclipse ide and look for all the possible output. Try to find all the possible solutions for this with proper logic.

Please try to finish it by tomorrow eod.

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Monday, September 07, 2015 3:12 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; Prakruti Mulimani; Sankar Theerthagiri; Bapu S Jadhav; KARTHICK SANTHARAM; Ravikiran Govindareddy; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik  
**Subject:** RE: Logical and analytical assignment 2  
**Importance:** High

Dear All ,

Please try to solve below problem statements by tomorrow eod:

1.Add arithmetic operators (plus, minus, times, divide) to make the following expression

true: 3 1 3 6 = 8. You can use any parentheses you’d like.

2. You have a 5 quart jug and 3 quart jug, and an unlimited supply of water (but no measuring

cups). How would you come up with exactly four quarts of water?

NOTE: The jugs are oddly shaped, such that filling up exactly ‘half’ of the jug

would be impossible.

3 .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 2 eggs. Find N, while

minimizing the number of drops for the worse case.

4. A bunch of men are on an island. A genie comes down and gathers everyone together

and places a magical hat on some people’s heads (e.g., at least one). The hat is magical:

it can be seen by other people, but not by the wearer of the hat himself. To remove the

hat, you must dunk yourself underwater at exactly midnight. If there are n people and

c hats, how long does it take the men remove the crowns? The men cannot tell each

other (in any way) that they have a hat.

FOLLOW UP

Prove that your solution is correct.

5 .There are 100 closed lockers in a hallway. A man begins by opening all the 100 lockers.

Next, he closes every second locker. Then he goes to every third locker and closes it if it

is open or opens it if it is closed (eg, he toggles every third locker). After his 100th pass in

the hallway, in which he toggles only locker number 100, how many lockers are open?

I am still waiting your responses for the Logical and analytical assignment 1.

Request to circulate this mail if  I have missed any name in the list .

Regards,

Purushottam

**From:** Purushottam Kumar(NABFS00)   
**Sent:** Wednesday, September 02, 2015 5:34 PM  
**To:** Sayan Bhattacharya; Sai Y Raghupatruni; Ravikiran Govindareddy; Susheel K Yadav; Narsa Reddy Mara; Anusha H Thimmappa; Fazalulhaq Shaik; Pavithra V; Shahid M Ghouse; Pawan K Rajak; Shabeena Khanum; Ravi K Korada; SyedaTahaseen Naaz; Varada Prasanth; Naveen K Kudari; Sadhu C Paikaray; Mohamed Shiraz; Taruna K Mohanty; Ashutosh Yadav; Smita B Hittalmani; Ajaya K Sahoo; Ajaya K Sahoo; Prabhakar K Jha; Siva Boda; Jeyasankar Arjunan; Varada Prasanth; Kundan K Sinha; Noor Afshan; Shreya Shridaran; Naveen K Kudari; SyedaTahaseen Naaz; Taruna K Mohanty; Prakruti Mulimani; Sankar Theerthagiri; Bapu S Jadhav; KARTHICK SANTHARAM; Ravikiran Govindareddy; Susheel K Yadav; Shahid M Ghouse; Diwaker Kalla; Bhavana A Rao; Ashutosh Yadav  
**Cc:** Arumugam Kanipakam; Satyaki Ranjan Ghosh; Asish Patnaik  
**Subject:** Logical and analytical assignment 1

Dear All ,

Please try to solve below problem statements by tomorrow eod:

1.Given two lines on a Cartesian plane, determine whether the two lines would intersect.

2.Given two squares on a two dimensional plane, find a line that would cut these two squares in half.

3.A circus is designing a tower routine consisting of people standing atop one

another’s shoulders. For practical and aesthetic reasons, each person must be

both shorter and lighter than the person below him or her. Given the heights

and weights of each person in the circus, write a method to compute the largest

possible number of people in such a tower.

EXAMPLE:

Input(ht wt) : (65, 100) (70, 150) (56, 90) (75, 190) (60, 95) (68,

110)

Output: The longest tower is length 6 and includes from top to bottom:

1. (56,90) (60,95) (65,100) (68,110) (70,150) (75,190).

4. Given a two dimensional graph with 6000 points on it, find a line which passes

the most number of points.

5. Suppose we have an array a1, a2, ..., an, b1, b2, ..., bn. Implement an algorithm

to change this array to a1, b1, a2, b2, ..., an, bn.

Please try to write

* **Algo.**
* **Logic**
* **Complete implementations of the problem .**

For the above mentioned all five problem statements.

Request to circulate this mail if  I have missed any name in the list .

Regards,

Purushottam