SET 1

MCQ

*1. Which of the following condition is required for a deadlock to be possible?(2 mark)*

a) mutual exclusion

b) a process may hold allocated resources while awaiting assignment of other resources

c) no resource can be forcibly removed from a process holding it

**d) all of the mentioned**

**Answer: d: all of the mentioned**

*2. Which one of the following is the deadlock avoidance algorithm?(2 mark)*

**a) banker's algorithm**

b) round-robin algorithm

c) elevator algorithm

d) karn's algorithm

**Answer: (A) Banker’s Algorithm**

*3. To avoid deadlock \_\_\_\_\_\_\_\_\_\_\_\_(2 mark)*

**a) there must be a fixed number of resources to allocate**

b) resource allocation must be done only once

c) all deadlocked processes must be aborted

d) inversion technique can be used

**Answer: (a) there must be a fixed number of resources to allocate**

*4. A Process Control Block(PCB) does not contain which of the following?(2 mark)*

a) Code

b) Stack

**c) Bootstrap program**

d) Data

**Answer: c) Bootstrap program**

*5. Which of the following is not the state of a process?(2 mark)*

a) New

**b) Old**

c) Waiting

d) running

**Answer: (b) Old**

**---------------------------------------------------------------------------------------------------------------------**

**Q. What is ACID properties and BASE properties? (10 mark)**

ACID Stands for

* Atomicity
* Consistency
* Isolation
* durability

A transaction is a single logicl unit of work which accesses logical unit of work which accesses and possibly modifies the content of a database

Inorder to maintain consistency in databse before and after transaction. Certain properties are allowed. These properties are called ACID properties.

**ATOMICITY**:- It simply meaning all or none.

In the concept of atomicity , if we have more than one transaction and if the transaction has failed before commit. Then we do roll back this transaction. And not execute any transaction process. It means it execute all transaction. Or none of transaction process**.**

**CONSISTENCY**:- It simply means before transaction starts and after the transaction completed. Does some of money should be same if some money of transaction is not same after the transaction then there is no consistency in the transaction.

**ISOLATION:\_** it means if more than one transaction are executed parallely then this time CPU moves upon them one by one now the situation we use this concept that we are converting these transaction in serial transaction.

It follows the conflict serialbility

**DURABILITY:-** Durability means when the change any kind of data in database then theses changes are permanent if changes are not permanent. Then there are no durability in transaction.

The **BASE** acronym is used to describe the properties of certain databases, usually NoSQL databases. It's often referred to as the opposite of ACID

Base stands for

**B**asically **A**vailable,

**S**oft state,

**E**ventual consistency

**Basically available** could refer to the perceived availability of the data. If a single node fails, part of the data won't be available, but the entire data layer stays operational.

**Soft state**: All I could find was the concept of data needing a period refresh. Without a refresh, the data will expire or be deleted.

* Automatic deletion of data in a database seems strange to me.
* Expired or stale data makes more sense. But this concept would apply to any type of redundant data storage, not just NoSQL. Does it describe something else then?

**Eventual consistency** means that updates will eventually ripple through to all servers, given enough time.

**Q. What are the differences between paging and segmentation? (10 mark)**

## Answer:- Paging

Paging can be defiend as a techniquw used to manage memory. Or we can say Paging is a memory management technique in which process address space is broken into blocks of the same size called pages. The size of the process is measured in the number of pages. Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called frames and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called frames and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.

**Segmentation**

The segementation can be defined as the technique used to segmenting memory spaces. Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program. When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory. Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a segment map table for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.

**Q. Explain the OOPS principles. (10 mark)**

### **Principles of OOP**

Object-oriented programming is based on the following principles:

* **Encapsulation**. The implementation and state of each object are privately held inside a defined boundary, or class. Other objects do not have access to this class or the authority to make changes but are only able to call a list of public functions, or methods. This characteristic of data hiding provides greater program security and avoids unintended data corruption.
* **Abstraction**. Objects only reveal internal mechanisms that are relevant for the use of other objects, hiding any unnecessary implementation code. This concept helps developers more easily make changes and additions over time.
* **Inheritance**. Relationships and subclasses between objects can be assigned, allowing developers to reuse a common logic while still maintaining a unique hierarchy. This property of OOP forces a more thorough data analysis, reduces development time and ensures a higher level of accuracy.
* **Polymorphism**. Objects can take on more than one form depending on the context. The program will determine which meaning or usage is necessary for each execution of that object, cutting down the need to duplicate code.

**Q. What will happen when you write www.google.com in your web browser (10 mark)**

**Answer:-** The browser is going to analyze the input. Usually if it has a ".com" it won't think you're typing search terms. And once it decides it must be a URL, it'll check that it has a scheme, if not, it'll add "http://" to the beginning. And since you didn't specify a number of HTTP protocol features, it'll assume defaults, like port 80, GET method and no basic auth.

Then it'll create an HTTP request and send that. In MAC address, TCP packet transfers, dropped packet handling. But anyway, a "google.com" DNS lookup will happen, and if it's not already cached a DNS service will reply with a list of IP addresses, because "google.com" doesn't just have a single IP address. Browsers will pick the first one by default

So the HTTP request jumps from node to node until it gets to the IP address of google.com Google would respond that you need to be using HTTPS - assuming with a 301 permanent redirect. So it would go all the way back to your browser, the browser would change the scheme to HTTPS, use the default 443 port and resend. This time the TLS handshake would take place between the load balancer and the browser client.

the next thing Google would do is put it through web application firewall rules on its load balancer to see if it's a malicious request. When it passes, the secure connection has probably been terminated and the google-side cached homepage will be returned in an HTTP response. Probably pre-zipped.

Google's response header would be read by the browser, cached according to the response header caching policy, then the body would be un-gzipped. And because it's google it's probably ultra-optimized: minified, likely a lot of pre-rendered content, inlined CSS, JavaScript and images to reduce network requests and the time-to-first-render. But that request will trigger a cascade of other requests, all concurrent because it should be running HTTP/2. While those requests are being made, JavaScript would be parsed, probably not blocking because they used the defer attribute on their tags - or async, I never did read about what those did individually.

But the browser has probably already rendered the search box and is working on the toolbar at the top, which is going to take some extra network requests - I probably already have a cookie or maybe local storage with an OAuth token - or maybe I'm using Chrome and it already knows who I am, and that request with auth gets sent to their Google+ API that tells the Google search page application who I am.

Another request would be sent to get my avatar image. At this point they've already browser-sniffed to see if I wasn't using Chrome, in which case they would have popped-in a tooltip to tell me that Chrome is awesome and I should be using that instead of anything