**Bangladesh Oil Gas and Mineral Corporation (Petrobangla)**
**Post Name: Assistant Manager (CSE/IT/ICT), Exam Date: 31/05/2024**
**Time: 09-10 AM, Non-Tech: 40*1.5 =60, Dept: 40 Marks**

| **1(a).** | **A C program is given. You have to write the output of the program when the value of a = 85.** |
|---|---|

If(a>=90) printf(“A”);
If(a>=80) printf(“B”);
If(a>=70) printf(“C”);
If(a>=60) printf(“D”);
Else printf (“F”)

**Output: BCD**

Analyzing the Code

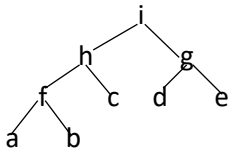
1. **Initial Condition Check (a >= 90)**:
   - When a = 85, the condition a >= 90 is false.
   - Therefore, printf("A"); will not execute.
2. **Second Condition Check (a >= 80)**:
   - The condition a >= 80 is true because 85 >= 80.
   - Therefore, printf("B"); will execute, and "B" will be printed.
3. **Third Condition Check (a >= 70)**:
   - The condition a >= 70 is true because 85 >= 70.
   - Therefore, printf("C"); will execute, and "C" will be printed.
4. **Fourth Condition Check (a >= 60)**:
   - The condition a >= 60 is true because 85 >= 60.
   - Therefore, printf("D"); will execute, and "D" will be printed.

| **1(b).** | **You have three access specifiers in java object oriented language. You have to find which access specifiers are worked with Public, Private and Protected Mode. If yes you have to right Y and If No you have to write N.** |
|---|---|

| | Within Class | Within Package | Outside package by sub class only | Outside Package |
|---|---|---|---|---|
| Public | Y | Y | Y | Y |

| | | | | |
|---|---|---|---|---|
| Private | Y | N | N | N |
| Protected | Y | Y | Y | N |

| | |
|---|---|
| **2.(a)** | **You are given a binary tree (a, b,c,d,e,f,g,h,i) nodes. The post order of the binary tree is:a b f c h d e g i nodes. Now draw the binary tree and show the array representation of this binary tree.** |




Array Representation: [a, b, c, d, e, f, g, h, i]

| | |
|---|---|
| **2(b)** | **In a quicksort algorithm taking first element as a pivot element. Now Analyze the time complexity of quicksort algorithm when all services of the quicksort algorithm are already sorted.** |

**Steps of Quicksort:**
Choose the first element as the pivot.
Partition the array such that elements less than the pivot are on the left and elements greater than or equal to the pivot are on the right.
Recursively apply the same procedure to the sub-arrays formed by partitioning.

**Worst-Case Scenario (Already Sorted Array)**
Partitioning Step
When the array is already sorted, choosing the first element as the pivot results in highly unbalanced partitions:

One sub-array will contain all the elements except the pivot.
The other sub-array will be empty.
For an array of n elements:

The first partitioning step will result in one sub-array of size n-1 and an empty sub-array.
The second partitioning step will result in one sub-array of size n-2 and an empty sub-array.
This process continues until the base case of a 1-element array is reached.

**Number of Comparisons**

During each partitioning step, the algorithm makes n-1 comparisons to partition the array:

First step: n-1 comparisons.
Second step: n-2 comparisons.
...
Last step: 1 comparison.
Summing these comparisons gives the total number of comparisons
(n-1) + (n-2)+…..+1 = $\frac{n(n-1)}{2}$
This summation is a well known arithmetic series.
Time Complexity: $\frac{n(n-1)}{2}$
$O(n^2)$

| | |
|---|---|
| **3** | **Employee (empno, PK, empname, monthlysalary, deptno, FK, mqrnd(FK))**<br>**Department(deptno, deptname,deptlocation)**<br>**Course(crscode,pk, crd desc, crs category, crs duration);**<br>**Offering(off begingate, crscode fk, offeringlocation, empno fk)** |

**(a) Dept has averageMonthlySalary greater than 1000;**

```
SELECT d.deptno, d.deptname, AVG(e.monthlysalary) AS average_salary
FROM Department d
JOIN Employee e ON d.deptno = e.deptno
GROUP BY d.deptno, d.deptname
HAVING AVG(e.monthlysalary) > 1000;
```

**(b) Course contain 2 course offering**

```
SELECT c.crscode, c.crsdesc, c.crscategory, c.crsduration, COUNT(o.crscode) AS
offering_count
FROM Course c
JOIN Offering o ON c.crscode = o.crscode
GROUP BY c.crscode, c.crsdesc, c.crscategory, c.crsduration
HAVING COUNT(o.crscode) = 2;
```

| | |
|---|---|
| **4(a)** | **A traffic signal cycles from GREEN to YELLOW, YELLOW to RED and RED to GREEN. In each cycle, GREEN is turned on for 70 seconds, YELLOW is turned on for 5 seconds and the RED is turned on for 75 seconds. This traffic light has to be implemented using a finite state machine (FSM). The only input to this FSM is a clock of 5 second period. The minimum number of flip-flops required to implement this FSM .** |

Green is turned ON for 70 seconds
Yellow is turned ON for 5 seconds.
Red is turned ON for 75 seconds
∴ Total time to complete one cycle for all 3 lights = (70 + 5 +75) seconds = 150 seconds
Available time period of clock = 5 seconds
Total number of clock cycles in one complete cycle = 150/5 = 30
Assume number of flip-flops required = n
We know possible number of states using n flip-flops = $2^n$
So to get 30 clock cycle or 30 states,
$2^n \geq 30$
$\Rightarrow n \geq 4.90$
∴ Minimum number of required flip-flops is = 5

| | |
|---|---|
| **4(b)** | **Given 5 scenario of a software engineering (Unit test, Regression Test, Smoke Test, Integration testing, Load Testing). One is done for you. Write the name of the testing and whether it is function/non function or both.** |

| | | |
|---|---|---|
| Verifying smallest part of a software system | Unit Testing | Functional |
| Re-running functional and non-functional tests to ensure that a software application still works as expected after changes. | Regression Testing | Functional |
| Testing specific amount of traffic | Load Testing | Non-Functional |
| Preliminary test of basic functional requirements | Smoke Testing | Functional |
| Verify if different modules or services work together as expected. | Integration testing | Functional |

| | |
|---|---|
| **5.(a)** | **What is the difference between Shell and kernel ?** |

Shell is the outer layer of OS. Kernel is the inner layer of OS.

Shell interacts with user and interprets to machine understandable language. Kernel directly interacts with the hardware by accepting machine understandable language from the shell.

Shell acts as an intermediary between the user and the kernel, Operates at a lower level than the shell and interacts with hardware.

| b | **How to solve when drive failure in RAID?** |
|---|---|

- **Identify the Failed Drive:** Your RAID controller software or BIOS should identify the failed drive. Look for error messages or warnings.
- **Power Down the System:** If possible, shut down the system to prevent further damage to the RAID array.
- **Consult Your RAID Documentation:** Refer to your RAID controller's manual for specific instructions on how to handle drive failures.
- **Replace the Failed Drive:** If your RAID level tolerates drive failures (RAID 1, 5, or 6), obtain a replacement drive with the same specifications (capacity, speed, etc.). Replace the failed drive according to the RAID controller's instructions.
- **Data Recovery (if necessary):** If data recovery is necessary (RAID 0 failure or multiple drive failures in RAID 5/6), consider professional data recovery services. They have specialized tools and techniques for recovering data from damaged RAID arrays.

| (c) | **What is the use of mmWaves in IoT?** |
|---|---|

IoT devices use mmWaves, as their high bandwidth capacity is ideal for applications such as short-distance wireless transmission of ultra-high-definition video and communications.

| (d) | **What is the need of an edge server?** |
|---|---|

An edge server is a computing system that's located at the edge of a network, close to data sources or end-users. Edge servers can act as an entry point into a network, connecting two or more distinct networks. They can also store cached versions of content and deliver it as quickly as possible.

Uses: Reduced Latency, Improved Bandwidth Efficiency, Enhanced Security, Offline Functionality, Scalability, Real-Time Insights: