**Question 1 –**

**import** java.util.\*;

**public** **class** ProgrammingTest\_Question1 {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***); // initializing Scanner to scan for inputs

System.***out***.println("The program prints the volume, surface area and circumference of a capsule\n having radius r ranging from 6 to N in increments of 6 and side length a");

System.***out***.print("Enter the value for N: ");

**int** valueN=sc.nextInt(); // Reading radius

System.***out***.print("Enter the length of side a: ");

**double** valueA=sc.nextDouble(); // Reading side

System.***out***.printf("%-12s\t%-10s\t%-12s\t%-12s\n","Radius","Volume","Surface Area","Circumference");

System.***out***.println("-------------------------------------------------------------");

**for** (**int** i = 6; i < valueN ; i += 6) //for loop to calculate the Volume, Surface area and Circumference of the capsule and display in the desired format

{

**double** Volume=Math.***PI***\*Math.*pow*(i,2)\*((4.0/3.0)\*i + valueA);

**double** SurfaceArea=(2\*Math.***PI***\*i)\*((2\*i)+valueA);

**double** Circumference=2\*Math.***PI***\*i;

System.***out***.printf("%-12d\t%-10.3f\t%-12.4f\t%-12.4f\n", i, Volume, SurfaceArea, Circumference);

}

}

}

**Question 2 –**

**import** java.util.\*;

**import** java.text.\*;

**public** **class** ProgrammingTest\_Question2 {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***); // initializing Scanner to scan for inputs

DecimalFormat decimalFormat = **new** DecimalFormat("#.00"); // Initializing Decimal format to round off dollars

decimalFormat.setGroupingUsed(**true**); // Enabling Dollar functionality for adding commas

decimalFormat.setGroupingSize(3); // Adding commas after 3 numbers

System.***out***.printf("Enter the number of projects: ");

**int** valueN = sc.nextInt(); // Stores the number of projects

System.***out***.println();

// Declaring arrays needed to store the project details

String projectName[] = **new** String[valueN];

Double upFrontCost[] = **new** Double[valueN];

Double rateOfReturn[] = **new** Double[valueN];

**int** duration[] = **new** **int**[valueN];

**double** cashflows[][] = **new** **double**[valueN][10]; // 2-D array to store yearly cashflows for each project

**double** totalIncome[] = **new** **double**[valueN];

**double** amount[] = **new** **double**[valueN];

**for** (**int** i = 0; i < valueN; i++) // for loop to take inputs for individual projects and store them in the arrays

// created.

{

System.***out***.printf("Enter the name of the project: ");

projectName[i] = sc.next();

System.***out***.printf("Enter the upfront cost for project %s: ", projectName[i]);

upFrontCost[i] = sc.nextDouble();

System.***out***.printf("Enter rate of return or discount rate(in %%): ");

rateOfReturn[i] = sc.nextDouble();

System.***out***.printf("Enter the duration(in years): ");

duration[i] = sc.nextInt();

**for** (**int** j = 1; j <= duration[i]; j++) // for loop to take individual year cashflows

{

System.***out***.printf("Enter the cash inflow-outflows during year " + j + ": ");

cashflows[i][j] = sc.nextDouble();

totalIncome[i] += cashflows[i][j];

}

System.***out***.println();

}

**for** (**int** i = 0; i < valueN; i++)// for loop to print out the table and details in the desired format

{

System.***out***.printf("\n\t\t\t %10s\n", projectName[i]);

System.***out***.println("--------------------------------------------------------------");

System.***out***.printf("%-8s|\t%-15s\t| %-8s | %-8s\n", "Year", "Cash", "PV Factor", "Amount");

System.***out***.printf("\t| %-20s|\n", "Inflows/Outflows");

System.***out***.println("--------------------------------------------------------------");

**for** (**int** j = 1; j <= duration[i]; j++) {

**double** PVFactor = 1.0 / Math.*pow*(1.0 + (rateOfReturn[i] / 100.0), j);

System.***out***.printf("%-8d|\t$%-15s| %-8.4f | $%-8s\n", j, decimalFormat.format(cashflows[i][j]), PVFactor,

decimalFormat.format((PVFactor \* cashflows[i][j])));

amount[i] += (PVFactor \* cashflows[i][j]);

}

System.***out***.printf("Total Income: $%s\n", decimalFormat.format(totalIncome[i]));

System.***out***.printf("Present Value of Future Benefits: $%s\n", decimalFormat.format(amount[i]));

System.***out***.printf("Present Value of Future Costs: $%s\n", decimalFormat.format(upFrontCost[i]));

System.***out***.printf("Net Present Value(NPV): $%s\n", decimalFormat.format(amount[i] - upFrontCost[i]));

}

// Variables declared to find out the maximum income and maximum NPV

**double** maxIncome = totalIncome[0];

**double** maxNPV = (amount[0] - upFrontCost[0]);

**int** maxIncomeCount = 0;

**int** maxNPVCount = 0;

**int** maxIncomeIndex[] = **new** **int**[2];

**int** maxNPVIndex[] = **new** **int**[2];

**int** k = 0, j = 0;

String maxIncomeString = "", maxNPVString = "";

**for** (**int** i = 1; i < valueN; i++) // for loop to find the maximum values

{

**if** (totalIncome[i] > maxIncome) {

maxIncome = totalIncome[i];

}

**if** ((amount[i] - upFrontCost[i]) > maxNPV) {

maxNPV = (amount[i] - upFrontCost[i]);

}

}

**for** (**int** i = 1; i < valueN; i++) // for loop to detect multiple maximum values

{

**if** (maxIncomeCount == totalIncome[i]) {

maxIncomeCount++;

maxIncomeIndex[j] = i;

j++;

}

**if** ((amount[i] - upFrontCost[i]) == maxNPV) {

maxNPVCount++;

maxNPVIndex[k] = i;

k++;

}

}

// Logic to unresolve maximum values in case there are multiple maximum values

**if** (maxIncomeCount > 1) {

**if** (amount[maxIncomeIndex[0]] - upFrontCost[maxIncomeIndex[0]] > amount[maxIncomeIndex[1]]

- upFrontCost[maxIncomeIndex[1]]) {

maxIncome = totalIncome[maxIncomeIndex[0]];

maxIncomeString = projectName[maxIncomeIndex[0]];

} **else** {

maxIncome = totalIncome[maxIncomeIndex[1]];

maxIncomeString = projectName[maxIncomeIndex[1]];

}

} **else** {

maxIncomeString = projectName[maxIncomeIndex[0]];

}

**if** (maxNPVCount > 1) {

**if** (totalIncome[maxNPVIndex[0]] > totalIncome[maxNPVIndex[1]]) {

maxNPV = (amount[maxNPVIndex[0]] - upFrontCost[maxNPVIndex[0]]);

maxNPVString = projectName[maxNPVIndex[0]];

} **else** {

maxNPV = (amount[maxNPVIndex[1]] - upFrontCost[maxNPVIndex[1]]);

maxNPVString = projectName[maxNPVIndex[1]];

}

} **else** {

maxNPVString = projectName[maxNPVIndex[0]];

}

// Print final output

System.***out***.println();

System.***out***.println("The Highest income is generated by project: " + maxIncomeString);

System.***out***.println("The project the company should be executing is: " + maxNPVString);

}

}

**Question 3 –**

*Query to create the source table -*

CREATE TABLE Employee (

Employee\_Id int,

Employee\_Name varchar(255),

Department varchar(255),

Year INT,

Vacation\_Days INT

);

*Query to insert data into the source table -*

INSERT INTO Employee (Employee\_Id, Employee\_Name, Department, Year, Vacation\_Days)

VALUES (00012, 'Luke Ye', 'Sales', 2011, 6),

(00013, 'Mark Brown', 'Marketing', 2012, 2),

(00016, 'James Tevlin', 'Engineering', 2011, 4),

(00017, 'Ross Becker', 'HR', 2012, 1),

(00012, 'Luke Ye', 'Sales', 2013, 2),

(00014, 'John Smith', 'Managment', 2011, 10),

(00013, 'Mark Brown', 'Marketing', 2012, 5),

(00016, 'James Tevlin', 'Engineering', 2012, 3),

(00017, 'Ross Becker', 'HR', 2013, 2),

(00017, 'Ross Becker', 'HR', 2012, 3),

(00015, 'Mark Brown', 'Marketing', 2013, 8),

(00012, 'Luke Ye', 'Sales', 2012, 1),

(00014, 'John Smith', 'Managment', 2011, 3),

(00015, 'Mark Brown', 'Marketing', 2014, 2);

*Query to create the first normalized table -*

CREATE TABLE Name\_Department (

Employee\_Name varchar(255),

Department varchar(255));

*Querying the normalized table -*

SELECT \* FROM Name\_Department;

*Query to create the second normalized table -*

CREATE TABLE Id\_Name\_year\_Vacation (

Employee\_Id int,

Employee\_Name varchar(255),

Year INT,

Vacation\_Days INT);

*Query to insert data into the first normalized table -*

INSERT INTO Name\_Department

SELECT Employee\_Name, Department from Employee GROUP BY Employee\_Name, Department;

*Query to insert data into the second normalized table -*

INSERT INTO Id\_Name\_year\_Vacation

SELECT Employee\_Id, Employee\_Name, Year, Vacation\_Days from Employee GROUP BY Employee\_Id, Employee\_Name, Year, Vacation\_Days;

*Query to join the normalized table and display the required output -*

SELECT nyv.Employee\_Id, nyv.Employee\_Name, nd.Department, nyv.Year, sum(nyv.Vacation\_Days)

FROM Id\_Name\_year\_Vacation nyv

LEFT JOIN Name\_Department nd

ON nd.Employee\_Name=nyv.Employee\_Name

GROUP BY nyv.Employee\_Id, nyv.Employee\_Name, nd.Department, nyv.Year

ORDER BY nyv.Employee\_Name, nyv.Year;