**Q1.pallindrome or not question**

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

String name = "radarr";

System.***out***.println( name.equals(**new** StringBuilder(name).reverse().toString())? "pallindrome ":"Not pallindrome");

junit

*assertTrue*("radar".equals(**new** StringBuilder("radar").reverse().toString()));

Q2. simple interest question

**public** **static** **double** calculate(**double** principal,**double** rate,**double** time) {

**return**(principal\*rate\*time)/100;

junit

*assertEquals*(200.0,myclass1.*calculate*(1000,5,4),0.01);

Q3.string reverse

**public** String reverse( String input) {

**if**(input == **null**) **return** **null**;

**return** **new** StringBuilder(input).reverse().toString();

junit

myclass3 junit = **new** myclass3();

*assertEquals*("tac",junit.reverse("cat"));

*assertNull*(junit.reverse(**null**));

*assertEquals*("",junit.reverse(""));

Q4.decimal to binary

**public** **static** String decimalToBinary(**int** decimal) {

**return** Integer.*toBinaryString*(decimal);

}

**junit**

int decimal = 10;

String expectedBinary = "1010";

String actualBinary = NumberConverter.decimalToBinary(decimal);

assertEquals(expectedBinary, actualBinary);

Q5.squarecube.

public static void main(String[] args) {

double number = 2.5; // Hardcoded input for testing without Scanner

double square = number \* number;

double cube = number \* number \* number;

System.out.println("Square of the number: " + square);

System.out.println("Cube of the number: " + cube);

junit

double number = 2.5;

double square = number \* number;

double cube = number \* number \* number;

assertEquals(6.25, square, 0.0001);

assertEquals(15.625, cube, 0.0001);

}

Q6.weeksdays

public static void main(String[] args) {

int days = 1000;

int years = days / 365;

int weeks = (days % 365) / 7;

int remainingDays = (days % 365) % 7;

System.out.println(days + " days is approximately:");

System.out.println(years + " years, " + weeks + " weeks, and " + remainingDays + " days.");

}

}

junit

int days = 1000;

int years = days / 365;

int weeks = (days % 365) / 7;

int remainingDays = (days % 365) % 7;

assertEquals(2, years);

assertEquals(38, weeks);

assertEquals(4, remainingDays);

}

Q7.factorial

**public static long calculateFactorial(int n) {**

**long factorial = 1;**

**for (int i = 1; i <= n; i++) {**

**factorial \*= i;**

**}**

**return factorial;**

**}**

**}**

**junit**

void testFactorialofn() { assertEquals(1, Factorial.calculateFactorial(0));

Q8. compare string

public void compareStrings(String str1, String str2) {

if (str1.equals(str2)) {

System.out.println("Strings are equal.");

} else {

System.out.println("Strings are not equal.");

}

}

public static void main(String[] args) {

String str1 = "Hello";

String str2 = "hello";

Stringcompare comparer = new Stringcompare();

comparer.compareStrings(str1, str2);

}

}

junit

String str1 = "Hello";

String str2 = "Hello";

assertEquals(str1,str2);

q9.voting

public static String checkEligibility(int age)

{

if (age >= 18) {

return "Eligible for voting";

}

else {

return "Not eligible for voting";

}

junit

assertEquals("Eligible for voting", vote.checkEligibility(18));