Practical Lab Week 2

Objective: the objective of this lab is to refresh your knowledge on Java programming.

1. Write a program that randomly generates an integer between 0 and 100 (0 and 100 are inclusive). The program asks the user to enter a number repeatedly until the number matches the randomly generated number. The program tells the user whether the input is too low or too high, so the user can guess the number closer to the right one at each input. Below shows a sample output:

```
Guess a magic number between 0 and 100
Enter your guess: 1
Your guess is too low
Enter your guess: 50
Your guess is too high
Enter your guess: 40
Your guess is too low
Enter your guess: 45
Your guess is too low
Enter your guess: 47
Your guess is too low
Enter your guess: 48
Your guess is too low
Enter your guess: 49
Yes, the number is 49
```

   Hint: use Scanner input = new Scanner(System.in); to read user input.

2. Write a program to print the following formatted output on the console.

```
Degrees    Radians    Sine       Cosine     Tangent
30         0.5236     0.5000     0.8660     0.5774
60         1.0472     0.8660     0.5000     1.7321
```

   Hint: use System.out.printf() to format the console output.

3. The following program was written to model a bank account that allows customers to withdraw and deposit funds.
   A. Identify all design and syntax errors in the program below (there are at least 5).
   B. Modify the withdraw method so that it tracks the number of withdrawals done, checks whether there is sufficient balance or not, and returns a boolean indicating the withdrawal succeeds or fails.
   C. Add another method to add interest based on the current interest rate supplied.

```
class Account
{
  private String name;
  private String ID;
  public double balance;

  public void Account(String name, String ID, double balance)
  {
    name = name;
    ID = ID;
```

```
        balance = balance;
    }

    public void getBalance()
    {
        return balance;
    }
    public void withdraw(double amt)
    {
        balance -= amt;
    }
    public void deposit(double amt)
    {
        balance += amt;
    }
}

public class TestAccount
{
    public static void main(String args[])
    {
        Account a1 = new Account("Tan A K", "S123", 24.5);
        Account a2 = new Account("Smith T", "S124", 1200.0);
        a1.deposit(100);
        a1.withdraw(2000);
        a2.deposit(120);
        a2.withdraw(80);
        System.out.println("Balance for " + a1.name + " is " + a1.getBalance());
        System.out.println("Balance for " + a2.name + " is " + a2.getBalance());
    }
}
```

4. Extend the account class to create a new subclass called CAccount for modelling a checking account, with the following features:
   A. Checking accounts can use an overdraft facility, so the new class will need instance variables for the overdraft limit and the amount overdrawn.
   B. The overdraft should not be used unless the current balance is insufficient to cover a withdrawal.
   C. When funds are deposited and there is an overdrawn amount, the funds should be used to reduce the overdrawn amount to 0 before the balance can be increased. For example, if overdrawn amount is $200, when $500 is deposited, overdrawn amount should be reset to 0, and the remaining amount $300 should be added to balance setting it at $300.
   D. A new constructor will be required that accepts the account name, ID, initial balance and overdraft limit. This constructor should use the super() facility to initialise the account ID, name and balance. This constructor should set the amount overdrawn to 0.
   E. Both withdraw and deposit methods must be overridden to accommodate these requirements.
   F. Create a CAccount object and invoke withdraw and deposit methods in TestAccount class.

5. Run the simple console program provided on Canvas and understand how it works.