PLACEMENT PREPRATION TEST
JAVA, C, DBMS

1) What could be the output?

```
  class A
 {
    int i;
    public static void main(String[] args)
    {
            System.out.println(i);
    }
 }
```

Ans:

2) What could be the output?

```
class X {
     X()
      {
       System.out.println("1");
      }
     static
      {
            System.out.println("2");
       }
     public static void main(String[] args)
     {
            new X();
      System.out.println("www.javapadho.com");
     }
 }
```

Ans:

3) What could be the output?

```
class Demo
{
   public static void main(String[] args) {
        int i = 0;
        int j = i++ + test(i) + i;
        System.out.println(i);
        System.out.println(j);
   }
   static int test(int i)
    {
      return i++;
   }
 }
```

Ans:

4) What could be the output?

```
 class Javapadho
  {
          public static void main(String[] args)
          {
                  int i = 10;
      int j = ++i + test1(++i) + test2(i);
      i=i-- + test1(i) + test2(i--)+i++;
      System.out.println(i--);
      System.out.println(--j);
  }
  static int test1(int i)
  {
          return ++i;
  }
  static int test2(int i)
  {
           return test1(i++);
  }
}
```

Ans:

5)

```
        public class JavaPadho{
public static void main(String[] args)
{
        String s1 = "SDJ";
        String s2 = "SDJ";
        String s3 = new String(s1);
        String s4 = new String(s2);
        System.out.println(s1 == s2);
        System.out.println(s3 == s4);
        System.out.println(s1 == s4);
        System.out.println(s3 == s4);
        System.out.println(s1.equals(s2));
        System.out.println(s1.equals(s3));
} }
```

ANS:

6.

```
public class T {
public static void main(String[] args)
{
        String s1 = "ja";
        String s2 = "va";
        String s3 = s1.concat(s2);
        String s4 = "java";
        System.out.println(s3 == s4);
}
}
```

ANS:

7)

```
public class Manager{
public static void main(String args[]){
String s1 = "---abc  123-hello---";
s1.replace("-", "  ");
s1.trim();
System.out.println(s1.length());
}
}
```

Ans:

8)

```
class Manager1
{
 public static void main(String []args)
        {
        System.out.println(test());
        }
private static int test()
{
        try
        {
                int i = 1000/(1000-1000);
                return 20;
        }
        catch(Throwable t)
        {
                return 0;
        }
        finally
        {
                return 1000;
        }
        return 2000;
}
```

Ans:

9)

```
public class K {
public static void main(String[] args)
{
        String s1 = "abc";
        StringBuffer sb = new StringBuffer(s1);
        sb.append("xyz");
        sb.append("hello");
        sb.append("test");
  System.out.println(sb);
}
}
```

Ans:

10.

```
public class Manager1
{
   static void test()
    {
     throw new RuntimeException("error occured");
    }
    public static void main(String[] args)
    {
        try
        {
         test();
        }
        catch(Error e)
        {
        System.out.println("caught");
} }}
```

Ans:

11.

```
class M
{
        static int x = test();
        static int test()
        {
                System.out.println("from test");
                return 10;
        }
        public static void main (String []args)
        {
          System.out.println(x);
        }
}
```

Ans

12.

```
import java.util.*;
public class JavaPadho {
public static void main(String[] args) {
    ArrayList list = new ArrayList();
    list.add("abc");
    list.add("aBC");
    list.add("ABC");
    list.add("abc");
    list.add("ABc");
    list.add("123");
    System.out.println(list);
    Collections.sort(list);
    System.out.println(list);
    list.set(2,"abc");
    System.out.println(list);

}
}
```

13.

```
public class Manager4
 {
    public static void main(String[] args)
    {
        Thread t1 = Thread.currentThread();
        System.out.println(t1.getId());
        System.out.println(t1.getName());
        System.out.println(t1.getPriority());
        System.out.println(t1.isDaemon());
    }
 }
```

Ans:

14.

```
class A1 extends Thread
   {
    public void run()
    {
        for(int i = 0; i < 10; i ++)
        {

            try
            {
                Thread.sleep(10);
            }
            catch(InterruptedException ex)
            {
                ex.printStackTrace();
            }

    System.out.println(i+":"+isInterrupted());
        }
    }
}
public class Manager3
{
    public static void main(String[] args)
    {
        A1 t1 = new A1();
        t1.start();
        t1.interrupt();

        System.out.println("done");
    }
}
```

15.

```
import java.util.ArrayList;
class A2
{    int i;
    A2(int i)
    {            this.i = i;
    }
}
public class M21
{
 public static void main(String[] args)
 {        ArrayList list = new ArrayList();
        list.add("abc");
        list.add(90);
        list.add(new A2(100));
        list.add(new A2(200));
        System.out.println(list);
 }
}
```

16.

```
#include<stdio.h>
```

```
int main()
{
          int a =5,b=6,c=7,d=8;
          printf(“%d%d%d”);
          return 0;
}
```

Ans:

17.

```
#include<stdio.h>
int main()
{
  int i;
  for(i=0;i<3;i++)
  {
    static int a=5;
   printf(“%d”,a++);
  }
return 0;
}
```

18.

```
#include<stdio.h>
int  main()
{
   int i,j=0;
   for(i=0;i<=3;++i, i++,++j)
   {
          printf(“%d%d”,i,j);
   }
return 0;
}
```

19.

```
void main()
{
    int a[3]={67,43,23};
    int *p = a;
    printf("%d",++*p);
    printf("%d",*++p);
    printf("%d",*p++);
}
```

Ans:

20.

```
int fun(int n,int *f_p)
{
   int t,f;
   if(n<=1)
   {    *f_p=1;
        return 1;
   }
   t =fun(n-1,f_p);
   f = t + *f_p;
   *f_p =t;
   return f;
}
int main()
{
   int x=15;
   printf("%d\n",fun(5,&x));
   getch();
   return 0;
}
```

21.

Which normal form is considered adequate for normal relational database design?

A) 2NF
B) 3NF
C) 4NF
D) 5NF

22.

Consider a schema R(A,B,C,D) and functional dependecies A→B and C→D. Then the decompsition of R into R1(AB) and R2(CD) is.

A) dependency preserving and lossless join
B) lossless join but not dependency preserving
C) dependency preserving but not lossless join
D) not dependency preserving and not lossless join

23.

Consider the following relation schema pertaining to a students database: Student (rollno,name,address) ,Enroll(rollno,courseno,coursename). Where the primary keys are shown underlined.The number of tuples in the student and Enroll tables are 120 and 8 respectively.What are the maximum and minimum number of the tuples that can be present in (Student * Enroll),where '*' denotes natural join ?

A) 8,8
B) 120,8
C) 960,8
D) 960,120

Ans:

24.

The relation schema student Performance (name,courseNo,rollNo,Grade) has the following functional dependencies:
name,courseNO→Grade , RollNo,CourseNo→Grade ,
Name→RollNo, RollNO →Name .

The highest normal form of this relation schema is.

A) 2NF
B) 3NF
C) 4NF
D) 5NF

25.

Consider a relation schema R=(A,B,C,D,E,H) on which the following functional dependencies hold:{A→B,BC→D,E→C,D→A}. What are the candidate keys of R ?

A) AE,BE
B) AE,BE,DE
C) AEH,BEH,BCH
D) AEH,BEH,DEH

Program:

1) Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:
1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

2) The prime factors of 13195 are 5, 7, 13 and 29.

What is the largest prime factor of the number 600851475143?

3) 2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder.

What is the smallest positive number that is evenly divisible by all of the numbers from 1 to 20?

4)

If the numbers 1 to 5 are written out in words: one, two, three, four, five, then there are 3 + 3 + 5 + 4 + 4 = 19 letters used in total.

If all the numbers from 1 to 1000 (one thousand) inclusive were written out in words, how many letters would be used?

NOTE: Do not count spaces or hyphens. For example, 342 (three hundred and forty-two) contains 23 letters and 115 (one hundred and fifteen) contains 20 letters. The use of "and" when writing out numbers is in compliance with British usage.

5)

The number, 197, is called a circular prime because all rotations of the digits: 197, 971, and 719, are themselves prime.

There are thirteen such primes below 100: 2, 3, 5, 7, 11, 13, 17, 31, 37, 71, 73, 79, and 97.

How many circular primes are there below one million?