PLACEMENT PREPRATION TEST ANS

1) CTE(Compile Time Error)
2) 2
   1
   www.javapadho.com
3) 1
   2
4) 46
   36
5) true

   false

   false

   false

   true

   true
6) False
7) 19
8) CTE (Unreachable code)
9) Abcxyzhellotest
10) Error occurred
    RTE
11) From test 10
12) [abc, aBC, ABC, abc, ABc, 123]
    [123, ABC, ABc, aBC, abc, abc]
    [123, ABC, abc, aBC, abc, abc]
13)

    1

    main

    5

    False
14) java.lang.InterruptedException: sleep interrupted

    done

    0:false

    1:false

15. [abc 90 hashcode hashcode]

16. 8 7 6

Default storage class of variable a, b, c and d is auto. Since it is automatic variable and Stored in the stack area of memory in the print function, name of variable is not written Explicitly .so default output will content of stack which will be in the LIFO order i.e 8 7 6.

17. 5 6 7

18. 0021

19. 68 43 43

20 . 8

21 . A

22 D

23 A

24 B

25 D

Problem 1

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.

Solution1:

```
package javapadho;

public class FibboEuler
```

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

        int num1 = 0;
        int num2 = 1;
        int temp = 0;
        int sum = 0;

        do
        {
          if (num2 % 2 == 0)
          {
            sum = sum + num2;
          }
          temp = num1 + num2;  //temp =1 2
          num1 = num2;  //num1 =1
          num2 = temp;  //num2 =2
        } while (num2 < 4000000);

        System.out.println(sum);
      }

}
```

Solution 2:

```
 public class FibboEuler
{
  public static void main(String[] args)
  {


      int sum = 0 ;
       int x1 = 1;
       int x2 = 2;
       while ( x1 < 4000000 )
       {
          if ( (x1 & 1) == 0 )  // x % 2 == 0
          {
            sum += x1;
```

```
            }
            x2=x1+x2;            // x2 = sum
            x1=x2-x1;            // x1 = the old value of x2
        }
        System.out.println(sum);
    }

}
```

2. Problem Statement

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

What is the largest prime factor of the number 600851475143?

```
public class LargestPrime
{

        public static void main(String[] args) {
         //   System.out.println(solve(600851475143l));
            System.out.println(solve(13195));
        }

        public static int solve(long number) {
            int i;

            // start dividing by 2, as that is the smallest prime number, keep going until we're trying to divide the number by itself,
            // this indicates we've finished.
            for (i = 2; i <= (number); i++) {
              //We're only interested in "i" if it divides evenly by the number
              if (number % i == 0) {
                // divide number by i and re-assign, so we can start counting up from 2 again
                number =number/ i;
                // the for loop will increment i, however if number is divisible by i with no remainder, we want to try again.
                // for example, if we divide number 2000 by 2, we want to decrement i so we can try to divide the resulting 1000 by 2 again
```

```
            i--;
        //      System.out.println(number);
          }
        }

        return i;
    }
}
```

Problem 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?

```
public class SmallestNumber {
public static long smallest() {
        long i = 2520;
        boolean found = false;
        while (!found) {
          i += 2520;
          boolean divis = true;
          for (int j = 11; j <= 20; j++) {
            if (i % j != 0) {
              divis = false;
             //  System.out.println(i + " is not divisible by " + j);
              break;
            }
            else {
              // System.out.println(i + " is divisible by " + j);
            }
```

```
            }
            if (divis) {
               found = true;
            }
          }
          return i;
        }
        public static void main(String[] args)
        {
              System.out.println(smallest());
        }
}
```

Solution 2:

```
public class SmallestNumber {
        static long FindLcm(long a,long b)
        {
           long lcm,hcf = 0;
           long i=1;
           long ger=a>b?a:b;
           while(i<ger)
           {
              if((a%i==0) && (b%i==0))
                 hcf=i;
              i++;
           }
           lcm=(a*b)/hcf;
           return lcm;
        }
        static void FindMultiple()
        {
           long lcm=1;
           for(long i=2;i<=20;i++)
```

```
            {
               lcm=FindLcm(lcm,i);
            }
            System.out.println("Lcm="+lcm);
         }
         public static void main(String[] args)
         {
                              FindMultiple();
        }

}
```

Ans**: Lcm=232792560**

Problem 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.

```
public class WordWritten
{
        public static String[] ones =
                    {
"","one","two","three","four","five","six","seven","eight","nine","ten",

"eleven","twelve","thirteen","fourteen","fifteen","sixteen","seventeen","eighteen",
"nineteen"};

                public static String[] tens =
```

```
{"","ten","twenty","thirty","fourty","fifty","sixty","seventy","eighty","ninety"};

                public static String[] hundreds =

{"","onehundred","twohundred","threehundred","fourhundred","fivehundred","si
xhundred",
                  "sevenhundred","eighthundred","ninehundred"};

                 public String word(int arg)
                {
                  String withAnd = "";

                  if(arg == 1000)
                  {
                    return "onethousand";
                  }

                  if(arg < 20)
                  {
                    return ones[arg];
                  }

                  if(arg < 100)
                  {
                       //divided arg by 10 gives it the correct index position in the
array of tens
                       //example: 80 is at position 8 in the tens array
                       //80/10 = 8. tens[8] = "eighty"
                    return tens[arg/10] + word(arg%10);
                  }

                    // if the number is not an even hundred value i.e 100, 200, 300
                    // then you have to add and to it according to the problem specs
                    //getting the remainder when divided by 100 allows you to see
                    //if the number needs an and
                    // ex: 124 % 100 = 24, so the answer would be
onehundredandtwentyfour
                  withAnd = word(arg % 100);
```

```
        if(withAnd.length() > 0)
          withAnd = "and" + withAnd;

          return hundreds[arg/100] + withAnd;
      }//end word

       public int letterCt(int arg)
       {
          System.out.println(word(arg));
          System.out.println(word(arg).length());
          return word(arg).length();
       }

       public static void main(String[] args)
       {
          int result = 0;

          for(int i = 1; i <= 1000; i++)
          {
             System.out.println("numWord: " + i);
             result += new WordWritten().letterCt(i);
          }

          System.out.println("Total Length: " + result);
       }
}
```

Ans:

```
numWord: 1
one
3
numWord: 2
two
3
numWord: 3
three
.
.
```

.
.
numWord: 999
ninehundredandninetynine
24
numWord: 1000
onethousand
11
Total length: 21224

Problem:

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?

Solution:

```
public class CircularPrime
{
        public static void main(String[] args) {
      int count = 0;
      int prime_count=0;

    for (long i = 2; i < 1000000; i++)
    {
       if(isprime(i))
       {
          prime_count++;
        String num=i+"";
        boolean a = false;
        String temp = num;
        for (int j = 0; j < num.length(); j++)
        {
           temp = temp.charAt(temp.length() - 1) + temp;
           temp = temp.substring(0, temp.length() - 1);
```

```
            if(!isprime(Long.parseLong(temp)))
            {
               a=false;
              break;
            }
            else
            {
               a=true;
            }
          }
            if (a)
            {
               count++;
            }
       }
        }
        System.out.println("prime count is : "+prime_count);
        System.out.println("THE ANSWER IS :" + count);
    }
    static boolean isprime(long num){

            boolean prime = true;
            if (num == 2) {
               prime = true;
            } else if (num % 2 == 0) {
               prime = false;
            } else {
               for (int j = 3; j <= Math.sqrt(num); j++) {
                  if (num % j == 0) {
                     prime = false;
                  }
               }
            }
      return prime;
    }
}
```