```
# shown for ssh
% ssh your_loginname@glados.cs.rit.edu
# type in your password
% cd where_your_solution_is_stored
% try hpb-grd lab1-4 JavaRunTimeVersion.java
Copying files...done

ON-TIME submission of lab1-4
Not Compiling your program...

Files being saved:
Board.java

lab1-4 has been submitted.
%
```

You can see if your submission was successful:

```
% try -q hpb-grd lab1-1
```

1.5. Teamwork

All work has to be submitted as a team of 2. You have to appear to the grading sessions on time. You have to select a grading slot during the first week. A schedule will be posted at the grad lab door at the beginning of the first week.

You will receive 0 points if you are late for your grading session.

The graders determine who answers the questions.

1.6. Homework 1.1 (10 Points)

Objective: Compilation of a Java program, designing, implementing, and testing of a algorithm.

Grading:
Correctness: You can lose up to 40% if your solution is not correct
Quality: You can lose up to 80% if your solution is poorly designed
Testing: You can lose up to 50% if your solution is not well tested
Explanation: You can lose up to 100% if your solution if you can not explain your solution during the grading session

Homework Description:

A prime number is defined as:

A number *p* is a prime number if *p* has the following properties:

- *p* must be a integer
- *p* > 1 and
- the factors of *p* are 1 and itself.

Look at the following program:

```
class Prime {

 public static boolean isPrime(int n) {

    for ( int index = 2; index < n; index ++ ) {
        if ( n % index  == 0 )
```

```
                                 return false;
                }

                return true;
          }
          public static void main( String args[] ) {
            for ( int index = 2; index <= 10; index ++ )
                if ( isPrime(index) )
                        System.out.println(index + " " );
          }
        }
```

```
Source Code: Src/21/Prime.java
```

This program prints out all prime numbers in the range of [2 ... 10].

The that every natural number *n* and *n* > 1, *n* is either a prime number, or can be represented as a product of prime numbers. In other words *n* can be represented as:

$$n = p_1 * \cdots * p_{k;}\ 1 \le i \le k$$

The sum of the prime factorials of *n* is defined as:

$$sum\_of\_prim\_factorials := \sum_{i=1}^{k} p_i$$

Explanation:
The prime factorials of 6 are 2 and 3. The sum of the prime factorials are 2 + 3 == 5.

Your Work:
Modify *Prime.java* in such a way that it prints out the sum of the prime factorials for 2 ≤ *n* ≤ 10.

Requirements:

- You have to name your program Prime.java
- You can only use basic types.
- You can not use any publicly available class or library which can determine if a number is a prime number.
- Your program has to compute the sum of the prime factorials. In other words your program can not be something like:

```
class PrimeWrong {

  public static void main( String args[] ) {
        System.out.println("The sum of all primes for 2:      2      (2)");
        System.out.println("The sum of all primes for 3:      3      (3)");
        System.out.println("The sum of all primes for 4:      4      (2 + 2)")
        System.out.println("The sum of all primes for 5:      5      (5)");
        System.out.println("The sum of all primes for 6:      5      (2 + 3)")
        System.out.println("The sum of all primes for 7:      7      (7)");
        System.out.println("The sum of all primes for 8:      6      (2 + 2 +
        System.out.println("The sum of all primes for 9:      6      (3 + 3)")
        System.out.println("The sum of all primes for 10:     7      (2 + 5)")
  }
}
```

```
Source Code: Src/21/PrimeWrong.java
```

Example:

An example of a solution execution:

```
% java Prime
The sum of all primes for 2:      2    (2)
The sum of all primes for 3:      3    (3)
The sum of all primes for 4:      4    (2 + 2)
The sum of all primes for 5:      5    (5)
The sum of all primes for 6:      5    (2 + 3)
The sum of all primes for 7:      7    (7)
The sum of all primes for 8:      6    (2 + 2 + 2)
The sum of all primes for 9:      6    (3 + 3)
The sum of all primes for 10:     7    (2 + 5)
```

Submission:

```
% ssh glados.cs.rit.edu # or use queeg.cs.rit.edu if glados is down
# password
# go to the directory where your solution is ...
% try hpb-grd lab1-1 'All files required'
# you can see if your submission was successful:
# try -q hpb-grd lab1-1
```

Solution:

(This solution serves as the basis for the discussion in class. Sometimes there will be errors introduced to show common mistakes)

```
/*
 * all is hardcoded
 */
class Prime {

  static String sumOfAllPrimesAsString = "";
  public static boolean isPrime(int n) {
        for ( int index = 2; index < n; index ++ ) {
                if ( n % index  == 0 )
                         return false;
        }

        return true;
  }
  private static int sumOfAllPrimes(int n) {
        int sumOfAllPrimes = 0;
        int mayBePrime = 1;
        if ( n < 2 )
                return -1;
        while ( n > 1 ) {
                mayBePrime++;
                if ( isPrime(mayBePrime)        ) {
                        while ( n % mayBePrime == 0 )   {
                                sumOfAllPrimes +=mayBePrime;
                                n = n / mayBePrime;
                                if ( sumOfAllPrimesAsString == "")
                                        sumOfAllPrimesAsString =  "" + mayBePrime;
                                else
                                        sumOfAllPrimesAsString =  sumOfAllPrimesAs
                        }
                }
        }
        return sumOfAllPrimes;
  }
  public static void main( String args[] ) {
        int minimum  = 2;
        int maximum  = 15;
        int sumOfAllPrimes = 0;


        for ( int index = minimum; index <= maximum; index ++ ) {
                sumOfAllPrimesAsString = "";
                System.out.println("The sum of all primes for " + index +
                                ":      " +
                                sumOfAllPrimes(index) + "       " +
                                " (" + sumOfAllPrimesAsString + ")" );
        }


  }
}
```

Source Code: Src/21_sol/Prime.java

1.7. Homework 1.2 (10 Points)

Objective: Designing, implementing, and testing of a algorithm.

Grading:
Correctness: You can lose up to 40% if your solution is not correct
Quality: You can lose up to 80% if your solution is poorly designed
Testing: You can lose up to 50% if your solution is not well tested
Explanation: You can lose up to 100% if your solution if you can not explain your solution during the grading session

Homework Description:

Given a set of sticks S, with | S | = *n*. The set of the lengths of the sticks are:

$$sl = \{s_{1,} \cdots s_n\}$$

Given is a stick *s_new*, with length *l*; does a combination of elemenets of

$$\{s_{1,} \cdots s_n\}$$

exist so such

$$l = \sum_{i=1}^{k} s_i ; 1 \le k \le n$$

If such a set exist print out one set.

Explanation:
Assume give is the following set S = { 1, 2, 3, 4, 6 } and *s_new*, with length *5*. The following combinations would add up to the lenght of 5:

- 1 + 4 = 5
- 2 + 3 = 5

Your Work:
A 1 inch, 2 inch stick would add up to a stick of 3 length, a 1 inch, 2 inch, and 2 inch stick could add up to a stick of length 5 and 4.

Your Work:
You have sticks with the following lengths: 1 inch, 5 inch, 8 inch, 12 inch, 12 inch, 35 inch, 35 inch, 35 inch, and 61 inch.

Write a program which can determine if a combination for f2s_new = 1, 6, 9, 24, 110, 111, 115, 62, 24, 202, 203, 204, 205 exist.

Write a program to solve the problem.

A snippet of the code might look like this:

```
public class Sticks {
    static int[] stickLengths = { 1, 5, 8, 12, 12, 35, 35, 35, 61 };
    static int[] unknowStickLengths = { 1, 6, 9, 24, 110, 111, 115, 62, 24, 202, 203, 204,
    ...
    public static void main( String[] arguments ) {
        for ( int index = 0; index < unknowStickLengths.length; index ++ )
                doTestLength(unknowStickLengths[index]);
    }
}
```

Requirements:

- You have to name the file: Sticks.java
- You can use arrays to store the stick lengths and unknowSticks lengths, and you have to use an iterative algorithm.
- You can hardcode all values in your program.
- You can use basic types and arrays.
- You can not use any publicly available class or library which can determine if a particular set of sticks matches the length.

Example:

An example of a solution execution:

```
% java Sticks
3 inch:        yes; used sickLengths = 3 inch
5 inch:        yes; used sickLengths = 5 inch
```

Submission:

```
% ssh glados.cs.rit.edu # or use queeg.cs.rit.edu if glados is down
# password
# go to the directory where your solution is ...
% try hpb-grd lab1-2 ’All files required’
# you can see if your submission was successful:
# try -q hpb-grd lab1-2
```

Solution:

(This solution serves as the basis for the discussion in class. Sometimes there will be errors introduced to show common mistakes)

```java
public class Sticks {
    static int[] stickLengths = { 1, 5, 8, 12, 12, 35, 35, 35, 61 };
    static int soManySticks = stickLengths.length;
    static int[] unknowStickLengths = { 1, 6, 9, 24, 110, 111, 115, 62, 24, 202, 2

    private static String theFollowingSticksAreUsed (int value) {
        String returnValue = "";
        for ( int index = soManySticks; index >= 0 ; index --)  {
                if ( ( ( 1 << index ) & value ) == ( 1 << index ) )
                        returnValue += stickLengths[index] + " inch ";

        }
        if ( returnValue == "" )
                returnValue = "empty set";
        return returnValue;
    }

    private static int calculteLengthForThisSet(int value) {
        int sum = 0;
        for ( int index = soManySticks; index >= 0 ; index --)    {
                if ( ( ( 1 << index ) & value ) == ( 1 << index ) )     {
                        sum += stickLengths[index];
                }

        }
        return sum;
    }
    private static void  doTestLength(int thisLength)   {
        int setSize         = (int)Math.pow(2,  stickLengths.length );
        boolean foundAset  = false;

        int index = 0;                  // see comment in loop

        while ( ( index < setSize ) && ! foundAset ) {
                int sum = calculteLengthForThisSet(index);
                if ( ! ( foundAset = ( thisLength == sum ) ) )
                        index ++;
        }
        if ( foundAset )        {
                System.out.println(thisLength + " inch: "        +
                                   "\tyes; used stickLengths = " +
                                   theFollowingSticksAreUsed(index) );
        } else
                System.out.println(thisLength + " inch: \tno");
    }
    public static void main( String[] arguments ) {
        for ( int index = 0; index < unknowStickLengths.length; index ++ )
                doTestLength(unknowStickLengths[index]);
    }
}
```

Source Code: Src/21_sol/Sticks.java