

## USER MANUAL FOR MAVEN

### CREATING MY FIRST MAVEN PROJECT

#### A. Using the command prompt

1. Make sure you are connected in the internet and already installed maven in your operating system.
2. Locate where you want to reside your project.
3. Type `mvn archetype:generate` to download the recent artifacts into your local repository and it will also to create a directory for your project to reside.
4. After downloading all the plugins, you may see the following issue: (*Choose a number or apply filter format: [groupId:]artifactId, case sensitive contains*): 7: ). Normally, the command line will give you hints on what you would type, in this case it gave number 7, so just press Enter.
5. Next is to fill up your **groupId**, this will identify your project uniquely and it should follow the package conventions. (e.g maven.demo)
6. Next is to fill up your **artifactID**, this is the name of the jar without version. (e.g my-app)
7. Next is to specify the current **version** of the archetype you are using. (e.g 1.0-SNAPSHOT)
8. Then, specify the name of your **package**.
9. Just press Enter afterwards, it means you confirmed the validation of your previous inputs. Then it's all set.

```
Define value for property 'groupId': maven.demo
Define value for property 'artifactId': my-app
Define value for property 'version' 1.0-SNAPSHOT: : 1.0-SNAPSHOT
Define value for property 'package' maven.demo: : maven.demo
Confirm properties configuration:
groupId: maven.demo
artifactId: my-app
version: 1.0-SNAPSHOT
package: maven.demo
Y: : Y
[INFO] -----
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.1
[INFO] -----
[INFO] Parameter: basedir, Value: C:\
[INFO] Parameter: package, Value: maven.demo
[INFO] Parameter: groupId, Value: maven.demo
[INFO] Parameter: artifactId, Value: my-app
[INFO] Parameter: packageName, Value: maven.demo
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\my-app
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 10:23 min
[INFO] Finished at: 2017-09-16T16:54:11+08:00
[INFO] Final Memory: 15M/167M
[INFO] -----
```

### STANDARD PROJECT STRUCTURE:

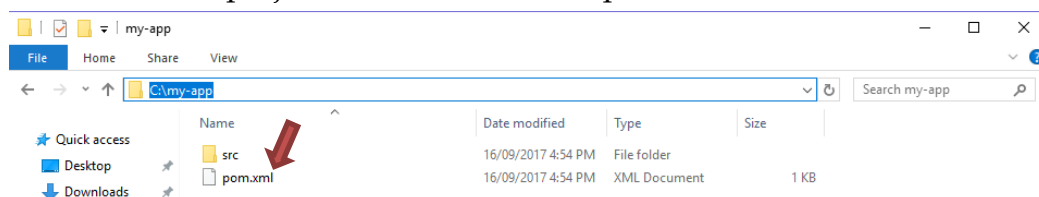
```
1. my-app
2. |-- pom.xml
3. `-- src
4.     |-- main
5.     |   |-- java
6.     |       |-- maven
7.     |           |-- demo
8.     |               |-- App.java
9. `-- test
10.     |-- java
11.     |   |-- maven
12.     |       |-- demo
13.     |           |-- AppTest.java
```

7.1 Or execute the following commands to give the same project structure shown above:

```
mvn -B archetype:generate\
-DarchetypeGroupId=org.apache.maven.archetypes\
-DgroupId=maven.demo
-DartifactId=my-app
```

### BUILDING JAVA CODES IN MAVEN

- a. You may have noticed an xml file on your maven-folder. The pom.xml file contains the project's name, version, dependencies and external libraries.



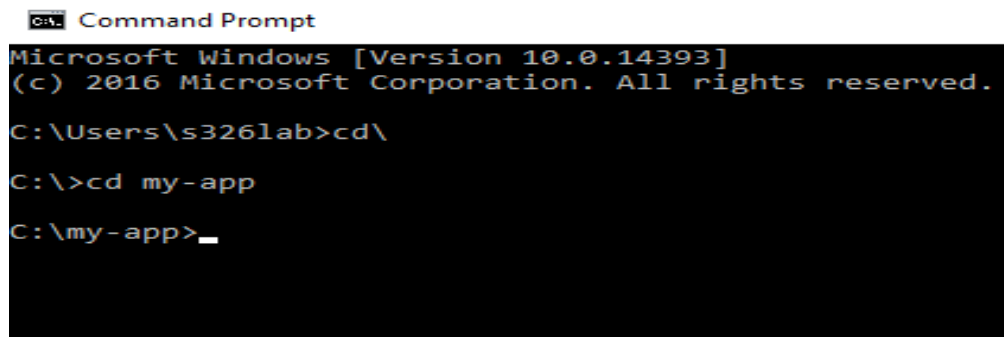
- b. You will need the following contents in your pom.xml in order to build your simple java project:

```
<?xml version="1.0"?>
- <project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xmlns="http://maven.apache.org/POM/4.0.0">
  <modelVersion>4.0.0</modelVersion>
  <groupId>maven.demo</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>my-app</name>
  <url>http://maven.apache.org</url>
  - <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  - <dependencies>
    - <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- c. Now, you are ready to build your project. The different commands to execute the build lifecycle goals (*validate*, *compile*, *test*, *package*, *verify*, *install*, *deploy*) in Maven are mentioned below:

*For compiling the code:*

1. Open the command prompt and locate where your maven project resides.



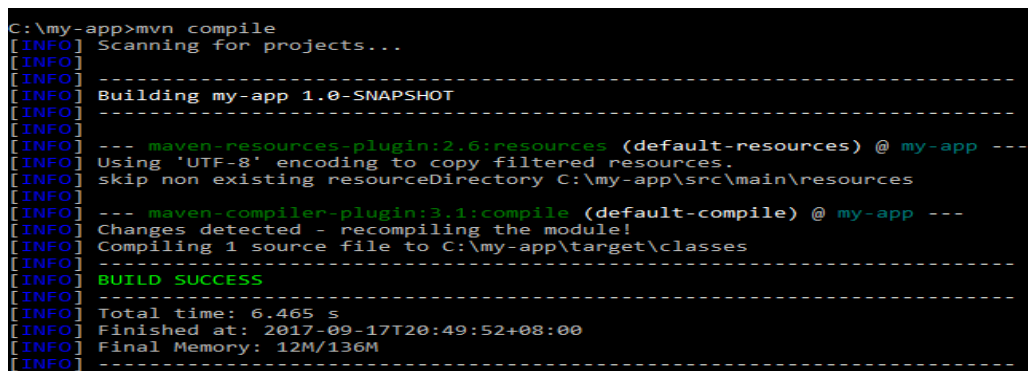
```
C:\> Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\s3261ab>cd \

C:\>cd my-app

C:\my-app>_
```

2. Type *mvn compile* to execute the compile goal.



```
C:\my-app>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\my-app\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ my-app ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\my-app\target\classes
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 6.465 s
[INFO] Finished at: 2017-09-17T20:49:52+08:00
[INFO] Final Memory: 12M/136M
[INFO] -----
```

3. This will create a target folder that contains the compiled class.

This PC > OS (C:) > my-app					Search my-app	
	<input type="checkbox"/>	Name	Date modified	Type	Size	
		pom.xml	9/17/2017 8:39 PM	XML Document	1 KB	
		target	9/17/2017 8:49 PM	File folder		
		src	9/17/2017 8:39 PM	File folder		

For testing the code:

1. Make sure you are in your Maven Project directory.
2. Type `mvn test` to run all the classes in your test folder.

```
C:\my-app>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\my-app\src\main\resources
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\my-app\src\test\resources
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ my-app ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\my-app\target\test-classes
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ my-app ---
[INFO] Surefire report directory: C:\my-app\target\surefire-reports

-----
T E S T S
-----
Running maven.demo.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.698 s
[INFO] Finished at: 2017-09-17T20:57:29+08:00
[INFO] Final Memory: 14M/192M
[INFO] -----
```

2.1 `mvn test` is for code coverage as well, you just have to put a certain plugin to execute it.

For creating jar file:

1. Open your command prompt and make sure you are in the directory where your project resides.
2. Type `mvn package` to compile your Java code, runs and validates tests and also packages the code into a jar file within the target directory.

```
C:\WINDOWS\system32\cmd.exe

C:\my-app>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\my-app\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\my-app\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ my-app ---
[INFO] Surefire report directory: C:\my-app\target\surefire-reports

-----
T E S T S
-----
Running maven.demo.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.016 sec

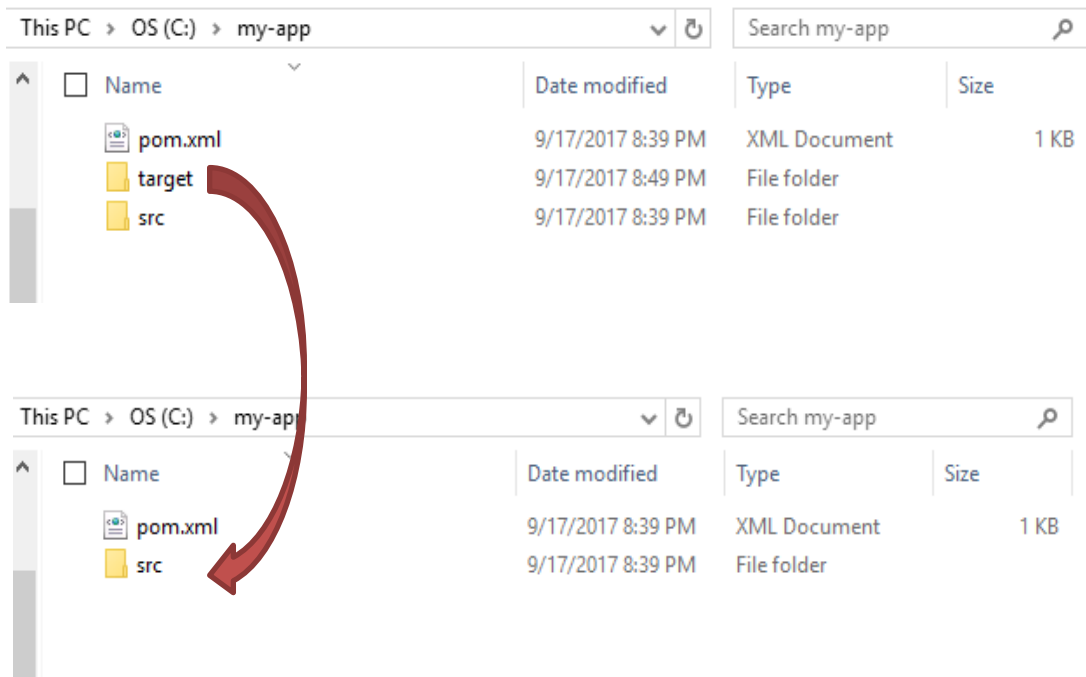
Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ my-app ---
[INFO] Building jar: C:\my-app\target\my-app-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 3.017 s
[INFO] Finished at: 2017-09-17T21:11:47+08:00
[INFO] Final Memory: 10M/232M
[INFO]
-----
```

To clean your files and directories:

1. Type *mvn clean* to clean up artifacts created by prior build.



## DECLARING DEPENDENCIES

Some applications may need external libraries in order for their program to compile because declaring dependencies in your *pom.xml* are scoped as compile dependencies. You may use *junit* as a dependency to compile and test your simple project:

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

So far we have learned the basics-- to compile, run tests, create a jar file. Now, in order for you to execute the project everytime the project is built, it requires certain plugins. Below is the sample pom.xml for compiling, running, creating jar files, code coverage as well as cleaning the project.

```
<build>
<plugins>
<!-- FOR CLEANING -->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-antrun-plugin</artifactId>
  <version>1.1</version>
<executions>
<!-- FOR COMPILING -->
<execution>
  <id>id.validate</id>
  <phase>validate</phase>
  <goals>
    <goal>run</goal>
  </goals>
  <configuration>
    <tasks>
      <echo>validate phase</echo>
    </tasks>
  </configuration>
```

```

        </execution>
<execution>
  <id>id.compile</id>
  <phase>compile</phase>
  <goals>
    <goal>run</goal>
  </goals>
  <configuration>
    <tasks>
      <echo>compile phase</echo>
    </tasks>
  </configuration>
</execution>
<execution>
  <id>id.test</id>
  <phase>test</phase>
  <goals>
    <goal>run</goal>
  </goals>
  <configuration>
    <tasks>
      <echo>test phase</echo>
    </tasks>
  </configuration>
</execution>
<execution>
  <id>id.package</id>
  <phase>package</phase>
  <goals>
    <goal>run</goal>
  </goals>
  <configuration>
    <tasks>
      <echo>package phase</echo>
    </tasks>
  </configuration>
</execution>
<execution>
  <id>id.deploy</id>
  <phase>deploy</phase>
  <goals>
    <goal>run</goal>
  </goals>

```

```

        <configuration>
            <tasks>
                <echo>deploy phase</echo>
            </tasks>
        </configuration>
    </execution>
<execution>
    <id>id.pre-clean</id>
    <phase>pre-clean</phase>
    <goals>
        <goal>run</goal>
    </goals>
    <configuration>
        <tasks>
            <echo>pre-clean phase</echo>
        </tasks>
    </configuration>
</execution>
<execution>
    <id>id.clean</id>
    <phase>clean</phase>
    <goals>
        <goal>run</goal>
    </goals>
    <configuration>
        <tasks>
            <echo>clean phase</echo>
        </tasks>
    </configuration>
</execution>
<execution>
    <id>id.post-clean</id>
    <phase>post-clean</phase>
    <goals>
        <goal>run</goal>
    </goals>
    <configuration>
        <tasks>
            <echo>post-clean phase</echo>
        </tasks>
    </configuration>
</execution>
</executions>

```



```

</plugin>
<!-- FOR CODE COVERAGE -->
<plugin>
  <groupId>org.jacoco</groupId>
  <artifactId>jacoco-maven-plugin</artifactId>
  <version>0.7.5.201505241946</version>
  <executions>
    <!-- Prepares the property pointing to the JaCoCo runtime agent which
    is passed as VM argument when Maven the Surefire plugin is executed. -->
    <execution>
      <id>pre-unit-test</id>
      <goals>
        <goal>prepare-agent</goal>
      </goals>
      <configuration>
        <!-- Sets the path to the file which contains the execution data. -->
        <destFile>${project.build.directory}/coverage-reports/jacoco-
ut.exec</destFile>
        <!-- Sets the name of the property containing the settings for JaCoCo
runtime agent. -->
        <propertyName>surefireArgLine</propertyName>
      </configuration>
    </execution>
    <!--Ensures that the code coverage report for unit tests is created after
unit tests have been run. -->
    <execution>
      <id>post-unit-test</id>
      <phase>test</phase>
      <goals>
        <goal>report</goal>
      </goals>
      <configuration>
        <!-- Sets the path to the file which contains the execution data. -->
        <dataFile>${project.build.directory}/coverage-reports/jacoco-
ut.exec</dataFile>
        <!-- Sets the output directory for the code coverage report. -->
        <outputDirectory>${project.reporting.outputDirectory}/jacoco-
ut</outputDirectory>
      </configuration>
    </execution>
  </executions>
</plugin>

```

```

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.15</version>
  <configuration>
    <!-- Sets the VM argument line used when unit tests are run. -->
    <argLine>${surefireArgLine}</argLine>
    <!-- Skips unit tests if the value of skip.unit.tests property is true
-->
    <skipTests>${skip.unit.tests}</skipTests>
    <!-- Excludes integration tests when unit tests are run. -->
    <excludes>
      <exclude>**/IT*.java</exclude>
    </excludes>
  </configuration>
</plugin>
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-failsafe-plugin</artifactId>
  <version>2.15</version>
  <executions>
    <!--Ensures that both integration-test and verify goals of the
Failsafe Maven plugin are executed. -->
    <execution>
      <id>integration-tests</id>
      <goals>
        <goal>integration-test</goal>
        <goal>verify</goal>
      </goals>
      <configuration>
        <!-- Sets the VM argument line used when integration tests are run. -->
        <argLine>${failsafeArgLine}</argLine>
        <!-- Skips integration tests if the value of skip.integration.tests
property is true -->
        <skipTests>${skip.integration.tests}</skipTests>
      </configuration>
    </execution>
  </executions>
</plugin>
</plugins>
</build>
</project>

```

----- Base Codes -----

```
package com.Mvn;
import java.util.*;

public class AllCodes {

    /* PROBLEM:  TOSMALLEST */
    /**
     *Algorithm:
     *<ul>
     *<li>1.Declare long minimum, that will be used to store
the lowest number possible
     *<li>2.Declare int index taht will hold the index of the
lowest number in n
     *<li>3.Declare int moveToIndex, that will hold the index
where you move the lowest number in n
     *<li>4.Declare String number that will hold the String
form of n
     *<li>5.for int i = 0; i is less than length of number;
increment i
     *<ul>
     *<li> 5.1.for int j=0; j is less than length of number1
increment j
     *<li> 5.1.1.if i is not equal to j making(number, i, j)
is less than minimum
     *<li> 5.1.1.1. assign making(number, i, j) to minimum
     *<li> 5.1.1.2. assign i to index
     *<li> 5.1.1.3. assign j to moveToIndex
     *</ul>
     *<li>6. return new long[]{minimum, index, moveToIndex}
     *</ul>
     *@param n is the number that you will need to rearrange
to have the lowest number by moving only one number
     *@return new long[] that will return the edited number,
index of the smallest number and the index where you the
smallest number
     */
    public static long[] smallest(long n) {
        long minimum = n;
        int index = 0;
        int moveToIndex = 0;
```

```

        String number = String.valueOf(n);
        for (int i=0; i<number.length(); i++) {
            for (int j=0; j<number.length(); j++) {
                if (i!=j && making(number, i, j) < minimum) {
                    minimum = making(number, i, j);
                    index = i;
                    moveToIndex = j;
                }
            }
        }
        return new long[]{minimum, index, moveToIndex};
    }
    /**
     *Algorithm:
     *<ul>
     *<li>1. Let sb be a StringBuilder and instantiate it
     *<li>2. Let c be a char that will hold the character at
given index of sb
     *<li>3. delete the character of sb at given index
     *<li>4. insert c in sb at given moveToIndex
     *<li>5.. Return Long.valueOf(sb.toString())
     *</ul>
     *@param number is the number to be edited
     *@param index is the index where the smallest number in
number is found
     *@param moveToIndex is the index where you move the
smallest number
     *@return value of the string
     */
    public static long making(String number, int index, int
moveToIndex) {
        StringBuilder sb = new StringBuilder(number);
        char c = sb.charAt(index);
        sb.deleteCharAt(index);
        sb.insert(moveToIndex, c);
        return Long.valueOf(sb.toString());
    }

    /* PROBLEM: TORTOISE */

    /**
     *Algorithm:
     *<ul>

```

```

* <li>1. Declare an int variable for hr, min , sec
* <li>2.1 If V1 is less than or equal to V2, return null
* <ul>
* <li>2.2 ELSE get the value of sec by subtracting
*   V2 to V1 (v2-v1) then divide it to the product
*   of 3600 and G (3600*g)    3600*g/(v2-v1)
* <li>2.3 get the value of hr by dividing the value of
*   second to 3600 (sec/3600)
* <li>2.4 get the value of the new sec by subtracting
*   the product of 3600 * hr (sec-3600*hr)
* <li>2.5 get the value of min by diving sec to 60
*   (sec/60)
* <li>2.6 get the value of sec by subtracting sec to
*   the product of 60 and min (sec - 60 * min)
* <li>2.7 return the new array of hr,min,sec {hr,min,sec }
* </ul>
* </ul>
* @param v1 is the integer velocity of tortoise A
* @param v2 is the integer velocity of tortoise B
* @param g is the integer lead of tortoise A
* @return int[] of time {hr,min,sec} of how long B will
catch A
**/
public static int[] race(int v1, int v2, int g) {
    int hr = 0;
    int min= 0;
    int sec = 0;
    if(v1>=v2)
    {
        return null;
    }
    sec = 3600 * g / (v2 - v1);
    hr = sec / 3600;
    sec= sec - 3600 * hr;
    min = sec / 60;
    sec = sec - 60 * min;
    System.out.print(new int[]{hr,min,sec});
    return new int[]{hr,min,sec};
}

/* PROBLEM: TANKTRUCK */

/**

```

```

*Algorithm:
*<ul>
*
* <li>1. Declare double radius equivalent to the
radius,divide the diameter by 2
* <li>2. Declare double radiusSquared equals to the value
of the radius squared
* <li>3. Declare double heightSquared equivalent to the
height squared
* <li>4. Declare double length equivalent to the maximum
volume of the tank divided by area of the cylinder top
* <li>5. Declare double equivalent answer equivalent to
the computed reamining value
* <li>6. Declare int result as answer
* <li>7. Return result
*</ul>
* @param h the height of the tank
* @param d the diameter of the tank
* @param vt the maximum volume of the tank
* @return result the remaining volume left in the tank
*/
public static int tankVol(int h, int d, int vt) {
    double radius=1.0f*d/2;
    double radiusSquared=Math.pow(radius,2);
    double heightSquared=Math.pow(h,2);
    double length=vt/(Math.PI*radiusSquared);
    double
answer=length*(radiusSquared*(Math.acos((radius-h)/radius))-
(radius-h)*(Math.sqrt(2*radius*h-heightSquared)));
    int result=(int)answer;
    return result;
}

```

```

/* PROBLEM: SUKOKU VALIDATOR */

```

```

/**
 *4 kyu Sudoku Solution Validator
 *
 *Algorithm:
 *<ul>

```

```

    * <li> 1. Construct a loop for row and column of the two
dimensional array
    * <li> 2. Declare a boolean variable and set it to true
    * <li> 3. Construct nested loops for the row and column
of the two dimensional array
    * <li> 4. If there is an equal value in one row or a
value that is equal to 0, set the variable to false
    * <li> 5. Return the variable

```

```

*/
* @param s integer of the 2-D array
* @return a the validated answer solution
*/
public static boolean check(int[][] s) {
    boolean a = true;
    for (int i = 0; i < s.length; i++) {
        for (int j = 0; j < s[i].length; j++) {
            for (int k = j+1; k < s[i].length-1; k++) {
                if (s[i][k] == 0) {
                    a = false;
                }
            }
        }
    }
    return a;
}

```

```

/* PROBLEM: SEQUENCES AND SERIES */

```

```

/**
 * Solution for: Sequences and Series by BattleRattle
 *
 * @param n the number to be tested
 * @return long the result of the tested number
 */
public static long getScore(long n) {
    return ((n*(n+1)*50)/2);
}

```

```

/* PROBLEM: MULTIPLES OF 3 AND 5 */

```

```

/**
 * 6 kyu Multiples of 3 and 5
 *

```

```

*Algorithm:
*<ul>
* <li>1. Declare an integer
* <li>2. Construct a loop that ends when the value of i
is greater
* or equal to the number.
* <li>3. Check if i%3 is equal to 0 or i%5 is equal to
zero
* 3.1 If true, sum = sum + i
* <li>4. Return sum
*</ul>
* @param number integer to be tested
* @return sum of the multiples of 3 and 5
*/

```

```

public static int solution(int number) {
    int sum = 0;
    for(int i = 0; i<number; i++){
        if(i%3==0 || i%5==0){
            sum += i;
        }
    }
    return sum;
}

```

```

/* PROBLEM: GAPINPRIMES */

```

```

/**
* <p>
* Finds the first pair of prime numbers that corresponds
to the given gap, and that has no prime numbers in between.
* </p>
*
* Algorithm:
* <ul style="list-style-type:none">
* <li>1. Start at the lower limit to the upper limit and
check if it is prime.
* <ul style="list-style-type:none">
* <li>1.1 If the number is prime then check if the
gap+1 is prime and if there's no prime numbers in between of
them.
* <ul style="list-style-type:none">

```



```

        *      <li>1.1.1 If there's none then return the
current pair.
        *      <li>1.1.2 Else, return null.
        *      </ul>
        *    </ul>
        *  </ul>
        *
        * @param gap, integer value of the wanted gap between to
primes.
        * @param lLimit, lower limit to which where to start the
search.
        * @param uLimit, upper limit to which where to stop the
search.
        * @return an array of long that has the value of the pair
prime number.
        */
        public static long[] firstGap(int gap, long lLimit, long
uLimit) {
            for(long i=lLimit; i<=uLimit; i++) {
                if(isPrime(i)) {
                    long j = gap + i;
                    if (isPrime(j) && !repeatedIsPrime(i + 1, j -
1)) {
                        return new long[]{i, j};
                    }
                }
            }
            return null;
        } //firstGap

/**
 * Checks if there's a prime number between a lower limit
and a higher limit.
 *
 * @param x, lower limit
 * @param y, upper limit
 * @return boolean value of whether there's a prime number
between x and y.
 */
        public static boolean repeatedIsPrime(long x, long y) {
            for(long i=x; i<=y; i++) {
                if(isPrime(i)) {
                    return true;
                }
            }
            return false;
        }

```

```

    }
}
return false;
} //repeatedIsPrime

/* PROBLEM: FOLDANARRAY */

/**
 *Algorithm
 *<ul>
 *<li>1. Declare an empty array dummy.
 *<li>2. While runs isn't equal to 0
 *
 *<li> 2.1 Declare an integer value length equivalent to
the length of the array
 * <li> 2.2 If the given array is divisible by two, equate
dummy to the length
 * <li> 2.2.1 Else equate dummy to length and add 1
 * <li>2.3 Divide the array into two, equate dummy to the
result of adding the values of the two halves
 *<ul>
 * <li>2.4 If length of the dummy array is equal to half
the length of the array
 *
 * <li> 2.4.1 Decrement runs by 1
 * <li> 2.4.2 Equate array to dummy
 * <li> 2.4.3 Return array
 *</ul>
 *</ul>
 *@param array use inputs an array
 *@param runs from a user input integer
 *@return array
 */
public static int[] foldArray(int[] array, int runs){
    int[] dummy=null;
    while(!(runs==0)){
        int length=array.length;
        if(array.length%2==0){
            dummy=new int[length/2];
        }else{
            dummy=new int[length/2+1];
        }
    }
}

```

```

        for(int first=0,last =array.length-1; first<last;
first++,last--){
            dummy[first]=array[first]+array[last];
        }
        if(array.length%2==1)dummy[dummy.length-
1]=array[length/2];
        runs--;
        array=dummy;
    }
    return array;
}

```

/\* PROBLEM: FLAMES \*/

```

/**
 *Algorithm:
 *<ul>
 *<li> 1. Declare initial variables that will hold for
the values of the names' combination,
 *      its result, and count.
 *<li> 2. For every input, it should be in lowercase form
 * <li> 3. Declare a variable copy to hold the original
letters of female input to use later
 * <li> 4. Use a for loop to check the length of the male
string
 * <li> 4.1 Replace all the common letters of female
string and male string to ""
 * <li>5. Use a for loop to check the length of the
female string (copy)
 * <li>5.1 Replace all the common letters of male string
and the original letters of female string (copy) to ""
 * <li>6. Add the combined result of the previous loop
and store it to the variable combined
 * <li> 7. Count the length of the result and store it to
the variable count
 * <li>8. Instead of cycling through all the letters, use
modulo to determine the remainder and use it to count.
 * <li>9. If result is greater than 6 or result is equal
to 0, equate result to 6
 *<li> 10. Checks the result in the following array of
strings
 *</ul>

```

```

    * @param male the string to be inputed for the male
    * @param female the string to be inputed for the female
    * @return the result for the flames
    **/
    public static String showRelationship(String male, String
female) {
        String combined = "";
        int result = 0;
        int count = 0;

        female = female.toLowerCase();
        male = male.toLowerCase();
        String copy = female;

        for(int x = 0; x < male.length(); x++){
            female =
female.replaceAll(String.valueOf(male.charAt(x)), "");
        }

        for(int x = 0; x < copy.length(); x++){
            male =
male.replaceAll(String.valueOf(copy.charAt(x)), "");
        }

        combined = male + female;
        count = combined.length();

        result = count%6;

        if(result == 0){
            result = 6;
        }

        String[] flames = {"Friendship", "Love", "Affection",
"Marriage", "Enemies", "Siblings"};
        return flames[result-1];
    }

    /* PROBLEM: AREWEALTERNATE? */
    /*
    *Algorithm:

```

```

*


*- 1. Check the first character if its a vowel
*- 2. Using a for loop check if every character is a
vowel

```

```

*
*@param word is the string to be checked
*@return boolean will tell if its alternate
**/

```

```

public static boolean isAlt(String word)
{
    boolean b = isVowel(word.charAt(0));
    for (int i = 1; i < word.length(); i++)
    {
        b = !b;
        if (b != isVowel(word.charAt(i)))
            return false;
    }
    return true;
}

static boolean isVowel(char ch)
{

```

```

    return "AEIOUaeiou".indexOf(ch) != -1;

```

```

}

```

```

/* PROBLEM: CREDITCARDVALIDATION */

```

```

/**

```

```

* <p>

```

```

* Puts the given string through a number of processes
which would then result into a boolean value
* that tells if it is a valid credit card number or not.

```

```

* </p>

```

```

*

```

```

* Algorithm:

```

```

* <ul style="list-style-type:none">

```

```

* <li>1. Get the numeric values of each character in cNum
and store it in nNum.

```

```

* <li>2. Starting from the furthest right and going left
by two's check if the doubled value of

```

```

    * nNum[j] is greater than 9.
    *   <ul style="list-style-type:none">
    *     <li>2.1. True, nNum[j] will now be equal to
nNum[j]*2-9.
    *     <li>2.2. False, nNum[j] will now be equal to
nNum[j]*2.
    *   </ul>
    * <li>3. Get the summation of all indexes of the array
nNum and store it ti tSum.
    * <li>4. Check if tSum%10==0.
    *   <ul style="list-style-type:none">
    *     <li>4.1. True, return true.
    *     <li>4.2. False, return false.
    *   </ul>
    * </ul>
    * @param cNum, String representation of the credit card
number to be processed.
    * @return Boolean value of whether the input cNum was
valid or not.
    */
    public static Boolean validate(String cNum) {
        int tSum = 0;
        int[] nNum = new int[cNum.length()];

        for(int i=0; i<nNum.length; i++) {
            nNum[i] = Character.getNumericValue(cNum.charAt(i));
        }

        for(int j=nNum.length-2; j>=0; j-=2) {
            nNum[j] = nNum[j]*2>9 ? nNum[j]*2-9 : nNum[j]*2;
        }

        for(int k=0; k<nNum.length; k++) {
            tSum += nNum[k];
        }

        return ((tSum%10)==0);
    } //validate

    /* PROBLEM: COUNTING DUPLICATES*/
    /**
     * <p>

```

```

        * This method returns how many characters are
duplicated.
    *</p>
    *Algorithm
    * <ul>
    * <li>1. Variable originaltext copies the originaltext
from the parameter
    * <li>2. Text is then converted to its lower case
    * <li>3. Create and initialize a stringbuilder to
copy the text
    * <li>4. Create and initialize a counter for a
character
    * <li>5. Create a loop that will check for the length
of the stringbuilder
    * <li>6. Create and initialize a char variable to
check the first character of the string.
    * <li>7. Check if the first character is not out of
range, then delete and increment the count variable
    * <li>8. If a character occurs more than once
increment the count variable else display none is repeated.
    * </ul>
    * @param text, a String to be determined by the
method
    * @return integer value of many characters are
repeated.
    */

```

```

public static int duplicateCount(String text) {

```

```

    String originaltext=text;
    text=text.toLowerCase();
    StringBuilder sb = new StringBuilder(text);
    int count2=0;
    while(sb.length() != 0)
    {
        int count = 0;
        char test = sb.charAt(0);
        while(sb.indexOf(test+"") != -1)
        {
            sb.deleteCharAt(sb.indexOf(test+""));
            count++;
        }
    }

```

```

        }
        if(count>1)
        {
            count2++;
            //System.out.println(originaltext+"->
"+count2+" "+test+" occurs "+count+" times");
        }

    }
    return count2;
}

/* PROBLEM: COUNT THE SMILEYS */

/**
 *Solution for Count The Smileys
 *Algorithm:
 *<ul>
 *<li>1. Check each index of the given array
 *<li>2. If the length of the string is 2
 *    <ul>
 *<li>2.1 Check if the index contains the ff: ":", ";", and
")", "D"
 *    <li>2.2 If yes, count + 1
 *    </ul>
 *<li>3. If the length of the string is 3
 *    <ul>
 *<li>3.1 Check each of the index if it contains the ff:
":", ";", "-", "~", "D"
 *    <li> 3.2 If yes, count + 1
 *    </ul>
 *    </ul>
 *@param arr list of the array that will contain the
strings to be counted
*@return number of the smileys given in the array */

public static int countSmileys(List<String> arr) {

    int c=0;
    for (String s : arr)
    {
        if (s.matches("[:;][-~]?[D]"))

```



```

        c++;
    }
    return c;
}

```

```

/* PROBLEM: BITCOUNTING */

```

```

/**
 * This method converts the inter into binary
 * counts all of the 1's in the binary
 * prints the number of 1's
 * @param n This is the number to be tested
 * @return int This returns the number of 1's in a binary
 */
public static int countBits(int n){
    return Integer.bitCount(n);
}

```

```

/* PROBLEM: BACKWARDSPRIME */

```

```

/**
 *Algorithm:
 *<ul>
 *<li>1.Let result be a String that will hold the result
of the method
 *<li>2.Let reversedPrime be a long that will hold the
reverse of the prime number
 *<li>3.long i = start; i less than or equal end; i++
 *<ul>
 *<li>3.1. if i is prime
 *<li>3.2. assign the reverse number of i to reversedPrime
 *<li>3.3. if reversedPrime is prime and i is not equal to
reversedPrime
 *<li>3.3.1. result+=i+"
 *</ul>
 *<li>4. return result.trim()
 *</ul>
 * @param start is the start of the number you will need
to find a prime number
 * @param end is the upper limit of the number you will
need to find a prime number
 * @return String result that is the concatitaion of all
the primes from start to end

```

```

        */
        public static String backwardsPrime(long start, long end)
        {
            String result = "";
            long reversedPrime= 0;
            for(long i = start; i <= end; i++){
                if(isPrime(i)){
                    reversedPrime = reverseNumber(i);
                    if(isPrime(reversedPrime) &&
!Long.toString(i).equals(new
StringBuilder(Long.toString(reversedPrime)).toString())){
                        result += i+" ";
                    }
                }
            }
            return result.trim();
        }
    }

```

```

/**
 *Algorithm:
 *<ul>
 *<li>1. if number modulo 2 is equal to 0
 *<li>1.1. return false
 *<li>     else
 *<li>2.for(int i = 3; i*i less than or equal number;
i+=2)
 *<ul>
 *<li>2.1.if number modulo i is equal to 0
 *<li>2.1.1. return false
 *</ul>
 *<li>3.return true
 *</ul>
 * @param number is the number that will be tested if it
is prime
 * @return boolean will return true if number is prime
else false
 */
    public static boolean isPrime(long number){
        if(number%2 == 0){
            return false;
        }else{
            for(int i = 3; i*i <= number; i+=2){
                if(number%i == 0){

```

```

        return false;
    }
}
return true;
}

/**
 *Algorithm:
 *<ul>
 *<li>1.Let reversedNum be a long that will hold the
reverse of num
 *<li>2.while num is not equal to 0
 *<ul>
 *<li>2.1.Let remainder be a long that will hold the value
of num modulo 10
 *<li>2.2.Assign reversedNum*10+remainder to reversedNum
 *<li>2.2.Assign num/10 to num
 *</ul>
 *<li>3.return reversedNum
 *</ul>
 *@param num is the number to be reversed
 *@return long reverseNumber is the reversed number of num
 */
public static long reverseNumber(long num){
    long reversedNum = 0;
    while(num != 0){
        long remainder = num%10;
        reversedNum = reversedNum*10+remainder;
        num = num/10;
    }
    return reversedNum;
}

/* PROBLEM: ROUTES */

/**
 * Solution for: 'Follow that spy' by adromil
 * Algorithm:
 *<ul>
 *    <li>1. Receive parameter value of routes
 *    <li>2. Declare String[] variable temp1, int
variable counter, String variable temp3, boolean

```

```

        *           variable spyFollowed, ArrayList(String[])
variable oldRoute, ArrayList(String) variable
        *           newRoute
        *           <li>3. Copy the contents of routes to oldRoute
        *           <li>4. Add the values of first array of oldRoute to
newRoute, then removes said array
        *           <li>5. While spyFollowed is false:
        *           <ul>
                *           <li> 5.1 Copy first array of oldRoute to temp1
        *           <li> 5.2 Compare temp1[1] String value to the
first String value of newRoute
        *           <li>5.3 If String values are equal:
        *           <li> 5.3.1 Insert temp1[0] value at index
0 of newRoute, remove array from oldRoute, and increment
counter by 1
        *           <li>5.3.2 Check if oldRoute is empty. If
empty, set spyFollowed to true
        *           <li>5.3 If String values are not equal,
increment counter by 1 and skip to next array
        *           <li> 5.4 If counter is equal to the size of
oldRoute (no match), proceed to switchRoute method
        *           </ul>
        *           <li> 6. Add all values of newRoute to temp3
        *           <li> 7. Return temp3
        * </ul>
        * @param routes the array of arrays that will contain the
strings to be sorted
        * @return String the sorted string
        */
    public static String findRoutes(String[][] routes) {
        String[] temp1; int counter=0; String temp3="";
        boolean spyFollowed = false;
        ArrayList<String[]> oldRoute = new
ArrayList<String[]>();
        ArrayList<String> newRoute = new ArrayList<String>();
// empty
        for (String[] temp2 : routes) {
            oldRoute.add(temp2);
        }
        temp1 = oldRoute.get(0);
        newRoute.add(temp1[0]);
        newRoute.add(temp1[1]);
        oldRoute.remove(0);

```

```

        if (oldRoute.isEmpty()) { spyFollowed = true; }
        while (!spyFollowed) {
            for (int a=0; a<oldRoute.size(); a++) {
                temp1 = oldRoute.get(a);
                if
(temp1[1].equalsIgnoreCase(newRoute.get(0))) {
                    newRoute.add(0, temp1[0]);
                    oldRoute.remove(a); counter=0;
                    if (oldRoute.isEmpty()) { // oldRoute
empty
                        spyFollowed = true;
                    }
                    break;
                } else if (counter == oldRoute.size()) {
// no match, oldRoute !empty
                    newRoute = switchRoute(newRoute,
oldRoute);

                    spyFollowed = true;
                    break;
                } else {
                    counter++;
                }
            }
        }
        temp3 = newRoute.get(0);
        for (int c=1; c<newRoute.size(); c++) {
            temp3 += ", " + newRoute.get(c);
        }

```

```

        return temp3;
    }

```

```

/**

```

```

 * Algorithm:

```

```

 * <ul>

```

```

 * <li> 1. Receives parameter values of newRoute and
oldRoute

```

```

 * <li> 2. Declare String[] variable temp4, boolean
variable switchDone

```

```

 * <li> 3. While switchDone is false:

```

```

 * <ul>

```

```

        <li> 3.1 Copy first array of oldRoute to temp4
    *        <li> 3.2 Compare temp4[0] to the last String
value of newRoute
    *        <li>3.3 If String values are equal:
    *            <li> 4.3.1 Insert temp4[1] value at the
end of newRoute and remove array from oldRoute
    *            <li> 4.3.2 Check if oldRoute is empty. If
empty, set switchDone to true
    *        </ul> 3.4 If String values are not equal, skip
to next array
    *        <li> 4. Return newRoute
    *</ul>
    * @param newRoute the ArrayList(String) for sorted
elements
    * @param oldRoute the ArrayList(String[]) for unsorted
elements
    * @return ArrayList(String) the sorted ArrayList(String)
    */
    public static ArrayList<String>
switchRoute(ArrayList<String> newRoute, ArrayList<String[]>
oldRoute) {
        String[] temp4; boolean switchDone = false;
        while (!switchDone) {
            for (int b=0; b<oldRoute.size(); b++) {
                temp4 = oldRoute.get(b);
                if
(temp4[0].equalsIgnoreCase(newRoute.get(newRoute.size()-1))) {
                    newRoute.add(temp4[1]);
                    oldRoute.remove(b);
                    if (oldRoute.isEmpty()) {
                        switchDone = true;
                        break;
                    }
                }
            }
        }
        return newRoute;
    }
}

```

