1. Write a Java program that counts down, one second at a time, from 10 to 1. After every second, your program should print the number of seconds remaining. Once your program has counted down to 1, print the string "Blast Off!". Do not output a new line until the countdown is finished. Your output should look like this:
10 9 8 7 6 5 4 3 2 1 Blast off!

Hints:
(a) Use both System.out.print() and System.out.println()
(b) Define and use the function
public static void sleep(long milliseconds), which returns after the length of time specified by the parameter milliseconds. An incomplete definition is given below.

```
public static void sleep(long milliseconds) {
    Date d; // This date object will be used to get the current time
    long start, now;
    // initialize d here by creating a new Date object
    /* d.getTime() returns the time at which d was initialized. Use d.getTime() to assign this time to "start" */
    do {
        /* Assign the current time to the variable "now". To get the current time, you need to first execute "d = new Date()". */
        } while ((YOU FILL IN THE BLANK HERE) < milliseconds);
}
```

2. Integer overflow occurs when an arithmetic operation attempts to create a numeric value that is too large to be represented within the available storage space. When this happens to a Java ”int” variable, which has only 32 bits of storage
space, its value becomes negative instead of increasing.
Write a simple while loop in which you iteratively increase the value of an "int" until you encounter overflow.
Hint: You might try doubling the value in each iteration of the loop until the result becomes negative

3. Floating point underflow occurs when an arithmetic operation attempts to create a numeric value that is too small to be represented within the available storage space. Write a simple while loop in which you iteratively divide a "float" variable by 2.0 until it is so small that its true value cannot be represented by a "float" and a value of 0 is used instead.

4. Write code that computes and outputs all prime numbers up to 20.

HINTS:
(a) To check if an integer $n$ is prime, compute the remainder of $n \div d$ for all $d < n$. If the remainder is equal to 0 for any one of these $d$, then $n$ is not prime. Otherwise, $n$ is prime.
(b) You will need to use a nested loop
(c) Use the $\%$ operator. $n \% d$ returns the remainder of $n/d$