- Create a program named `Challenges`. Include both your name and your partner's name (if you are working with someone) in a comment.
- Hand in your program at the end of class by copying it to your handin folder `/classes/cs124/handin/`*`username`* – one handin per group (make sure your names are in a comment!).

For each of the following, also write Javadoc comments, identify and check preconditions, and add code to `main` to test your functions and subroutines for the examples given.

- Write a function `blackjackSum` which, given two numbers 1-13 as parameters, returns the sum of the numbers unless one or both of the numbers is 11 (in which case the 11s are replaced with 1s in the sum) or the sum is greater than 21 (in which case 0 is returned). For example, the blackjack sum of 4 and 11 is 5 (replace the 11 with 1), the blackjack sum of 10 and 13 is 0 (10+13 > 21), and the blackjack sum of 5 and 8 is 13.
- Write a function `divides` which takes a number and a divisor (both integers > 0) as parameters and returns true if the divisor evenly divides the number and false otherwise. For example, the divisor 5 evenly divides 20 but the divisor 3 does not.

  Hint: *b* evenly divides *a* if *a* % *b* == 0
- Write a function `sumMultiples` which takes a limit (an integer) as a parameter and returns the sum of all of the multiples of 3 or 5 below that limit. For example, the sum of the multiples of 3 or 5 less than 10 is 22, less than 100 is 2632, and less than 1000 is 266332. Use your `divides` function to determine if a number is a multiple of 3 or a multiple of 5.
- Write a function `equalSplit` which, given an array of integers as a parameters, determines whether it is possible to split the array into two parts so that the numbers in each part sum to the same value. Return the index of the start of the second part if the split is possible and -1 otherwise. For example, for { 2, 3, 5, 7, 12, 4, 1 } it is possible to split – 2+3+5+7 = 12+4+1, so 4 (the index of 12) should be returned. For { 1, 2, 3, 4, 5, 6 } it isn't possible to split so -1 should be returned.

  Hint: first sum the numbers in the array to find the total, then go through the elements again – if twice the sum of elements A[0..i] equals the total, i+1 is the desired index. For example, the sum of the elements in { 2, 3, 5, 7, 12, 4, 1 } is 34 and the sum of A[0..3] (the elements 2, 3, 5, 7) is 17 so 3+1 = 4 is the index to return.
- Write a function `isPrime` which takes an integer (≥ 2) as a parameter and returns true if the number is prime and false otherwise. (A prime number is only divisible by itself and 1.) For example, 2, 3, 5, 7, 11 are all prime; 4, 6, 8, 9, 10 are not.

  Hint: a simple way to test if a number *n* is a prime is to try dividing it by everything from 2 to *n*-1. Use your `divides` function to test divisibility.
- Write a function `largestPrimeFactor` which takes an integer (≥ 2) as a parameter and returns the largest prime factor, that is, the largest prime number that evenly divides the number. For example, the largest prime factors of 10, 403, and 207900 are 5, 31, and 11, respectively.

  Hint: a simple strategy for finding the largest prime factor of *n* is to count from *n* down to 2, checking if the current factor is prime and evenly divides *n* (use your `isPrime` and `divides` functions) – if so, it is the largest prime factor.
- Write a function `printFib` which takes an integer *n* (≥ 1) and prints the first *n* Fibonacci numbers. (Print them on one line, separated by spaces, with a newline but no final space after the last number.) The Fibonacci sequence is 1, 1, 2, 3, 5, 8, 13, 21, 34, … – the first two numbers are 1 and 1, then subsequent numbers are computed by adding the previous two together (1+1 = 2, 1+2 = 3, 2+3 = 5, 3+5 = 8, etc).

  Hint: `printFib(1)` – 1 – and `printFib(2)` – 1 1 – are special cases, then have two loop variables for the last two numbers – use those to compute and print the next number in the sequence, then update them.
- Write a function `fibSum` which takes a limit and a divisor (both integers ≥ 1) and returns the sum of the Fibonacci numbers no greater than the limit and evenly divisible by the divisor. For example, the sum of the even Fibonacci numbers up to 40 is 44, the divisible-by-5 numbers up to 100 is 60, and the divisible-by-3 numbers up to 120 is 24.