Herhalingsoefeningen

Oefening 1: FizzBuzz

Write a method that returns 'Fizz' for multiples of three and 'Buzz' for the multiples of five.
For numbers which are multiples of both three and five return 'FizzBuzz'.
For numbers that are neither, return the input number.

Use this method to print all the numbers from 1 to 100, replacing each multiple of 3 by the word "Fizz", each multiple of 5 by the word "Buzz" and each multiple of 3 and 5 by the word "FizzBuzz". The output should look like this:

```
1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14,
FizzBuzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26,
Fizz, 28, 29, FizzBuzz, 31, 32, Fizz, 34, Buzz, Fizz, ...
```

Extra

Rewrite your program using Streams.

Oefening 2: Palindrome check

A palindrome is a word that reads the same backward or forward. '**abcba**' is a palindrome.
Write a method that detects if a string is a palindrome. The method should be case insensitive.
Use word.charAt(i) to get the character at position i.

Write a program that asks the user for input and tells the user if their input is a palindrome or not.

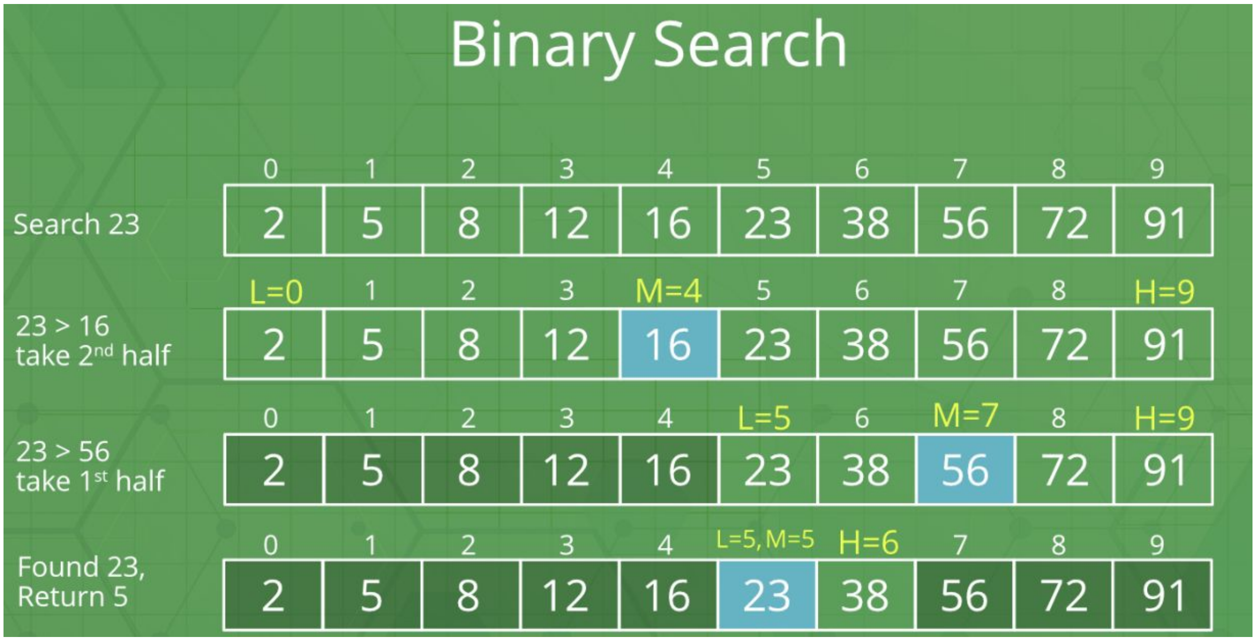
Oefening 3: Leap Year

Write a method that calculates if a given year is a leap year according to the following definition:

Every year that is exactly divisible by four is a leap year, except for years that are exactly divisible by 100, but these centurial years are leap years if they are exactly divisible by 400. For example, the years 1700, 1800, and 1900 are not leap years, but the years 1600 and 2000 are.

Oefening 4: Binary Search

Binary Search: Search a **sorted** array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

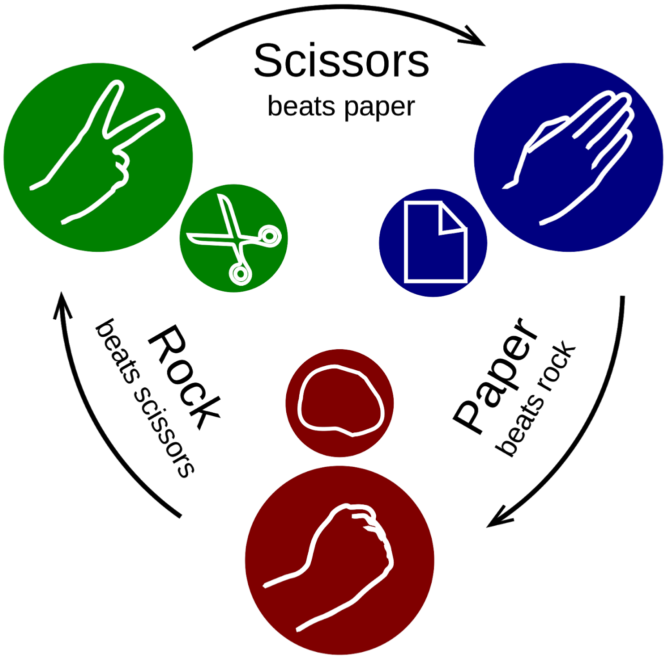


Write a method that takes in a **sorted** array of integers and a value to search for. Use binary search to find the index of the number to search for and return the index if the number can be found. Return -1 if the number to find is not in the array.

Oefening 5: Rock-paper-scissors

Write a program that asks the user for their move (can either be rock, paper or scissors). Let the program pick a move at random and compare the two moves to determine the outcome of the game:

User	**Program**	**Outcome**
`Rock`	`Rock`	`Tie!`
`Rock`	`Paper`	`I win!`
`Rock`	`Scissors`	`You win!`



Oefening 6: To-do list

Write a command line program that allows a user to create a todo list using text commands. It should support at least the following commands:

- `list`
- `add`
- `remove <number>`
- `edit <number>`
- `help`
- `exit`

Each todo item has a priority: `HIGH, NORMAL, LOW`

Commands

List

Print the current list of todo’s to the console, ranked by priority from highest to lowest:

```
TODO:
   1. Finish todo list app (HIGH)
   2. Drink plenty of water (HIGH)
   3. Go shopping (NORMAL)
   4. Do laundry (NORMAL)
   5. Water plants (LOW)
```

Print out: `You have no todo’s!` If there are no items in the list.

Add

Ask the user for additional input:

- Description of the todo item to add
- Priority (`HIGH, NORMAL, LOW`)

Remove <number>

Remove the todo item at the given number. Optionally ask the user for confirmation before removing the item.

Edit <number>

- Display the todo item at the given number
- Allow the user to replace the item description or change its priority
- Display the list of items after editing is done

Help

Display the list of possible commands

Exit

Close the program

Extra

- Add an extra command `done <number>`, to mark todo’s as done
  - Display todo’s that are done in a second list, labeled `DONE:`
- Save the todo list to a file on exit and reload them when the program starts