```
// Type definitions for [~THE LIBRARY NAME~] [~OPTIONAL VERSION NUMBER~]
// Project: [~THE PROJECT NAME~]
// Definitions by: [~YOUR NAME~] <[~A URL FOR YOU~]>

/*~ If this library is callable (e.g. can be invoked as myLib(3)),
 *~ include those call signatures here.
 *~ Otherwise, delete this section.
 */
declare function myLib(a: string): string;
declare function myLib(a: number): number;

/*~ If you want the name of this library to be a valid type name,
 *~ you can do so here.
 *~
 *~ For example, this allows us to write 'var x: myLib';
 *~ Be sure this actually makes sense! If it doesn't, just
 *~ delete this declaration and add types inside the namespace below.
 */
interface myLib {
    name: string;
    length: number;
    extras?: string[];
}

/*~ If your library has properties exposed on a global variable,
 *~ place them here.
 *~ You should also place types (interfaces and type alias) here.
 */
declare namespace myLib {
    //~ We can write 'myLib.timeout = 50;'
    let timeout: number;

    //~ We can access 'myLib.version', but not change it
    const version: string;

    //~ There's some class we can create via 'let c = new myLib.Cat(42)'
    //~ Or reference e.g. 'function f(c: myLib.Cat) { ... }
    class Cat {
        constructor(n: number);

        //~ We can read 'c.age' from a 'Cat' instance
        readonly age: number;

        //~ We can invoke 'c.purr()' from a 'Cat' instance
        purr(): void;
    }

    //~ We can declare a variable as
    //~   'var s: myLib.CatSettings = { weight: 5, name: "Maru" };'
    interface CatSettings {
        weight: number;
        name: string;
        tailLength?: number;
    }

    //~ We can write 'const v: myLib.VetID = 42;'
    //~  or 'const v: myLib.VetID = "bob";'
    type VetID = string | number;
    //~ We can invoke 'myLib.checkCat(c)' or 'myLib.checkCat(c, v);'
    function checkCat(c: Cat, s?: VetID);
}
```