persons.test.js

```javascript
import {describe, it, expect, beforeEach} from 'vitest';
import * as persons from "../src/persons.js";


const names = [
  "addIsAdultProperty",
  "addToAll",
  "adultFilter",
  "ageAverage",
  "average",
  "extractAge",
  "hasChild",
  "isAdult",
  "isAllAdult",
  "isChild",
  "sum",
];
it.each(Object.keys(persons))(
  '"%s" is an expected function name',
  (name) => expect(names).toContain(name),
);

if (typeof persons.sum === "function") {
  describe("sum", () => {
    it("sum of 1, 2 and 3 equals 6", () => {
      expect(persons.sum([1, 2, 3])).toBe(6);
    });
    it("sum of an empty array equals 0", () => {
      expect(persons.sum([])).toBe(0);
    });
  });
}

if (typeof persons.addToAll === "function") {
  describe("addToAll", () => {
    it("add 2 to all : 1, 2 and 3 equals 3, 4, 5", () => {
      expect(persons.addToAll([1, 2, 3], 2)).toEqual([3, 4, 5]);
    });
    it("return a new array", () => {
      const values = [1, 2];
      expect(persons.addToAll(values,0)).not.toBe(values);
    });
  });
}

if (typeof persons.average === "function") {
  describe("average", () => {
    it("of an empty array equals null", () => {
      expect(persons.average([])).toBe(null);
    });
    it("of -1, 2 and 5 equals 2", () => {
      expect(persons.average([-1, 2, 5])).toBe(2);
    });
  });
}

describe("user", () => {
  let child;
  let adult;
  if (
    typeof persons.isAdult === "function" ||
    typeof persons.isChild === "function"
  ) {
    beforeEach(() => {
      child = { age: 12 };
      adult = { age: 22 };
    });
  }

  if (typeof persons.isAdult === "function") {
    it("Child user is not an adult", () => {
      expect(persons.isAdult(child)).toBe(false);
    });
    it("Adult user is an adult", () => {
      expect(persons.isAdult(adult)).toBe(true);
    });
  }
  if (typeof persons.isChild === "function") {
    it("Child user is a child", () => {
      expect(persons.isChild(child)).toBe(true);
    });
    it("Adult user is not a child", () => {
      expect(persons.isChild(adult)).toBe(false);
    });
  }
});

describe("users", () => {
  let users;
  if (
    typeof persons.extractAge === "function" ||
    typeof persons.adultFilter === "function" ||
    typeof persons.ageAverage === "function" ||
    typeof persons.isAllAdult === "function" ||
    typeof persons.hasChild === "function" ||
    typeof persons.addIsAdultProperty === "function"
  ) {
    beforeEach(() => {
      users = [
        { age: 19, name: "Alice" },
        { age: 12, name: "Bob" },
        { age: 21, name: "Jim" },
        { age: 16, name: "John" },
        { age: 32, name: "Kelly" },
      ];
    });
  }

  if (typeof persons.extractAge === "function") {
    describe("extract age", () => {
      it("Extract users age in a new array", () => {
        expect(persons.extractAge(users)).not.toBe(users);
      });
      it("Extract users age return an array of users' age", () => {
        expect(persons.extractAge(users)).toEqual([19, 12, 21, 16, 32]);
      });
    });
  }

  if (typeof persons.adultFilter === "function") {
    describe("adultFilter", () => {
      it("filter without type return all data array", () => {
        expect(persons.adultFilter(users)).toEqual(users);
      });
      it('filter with "child" return child users', () => {
        expect(persons.adultFilter(users, "child")).toEqual([
          { age: 12, name: "Bob" },
          { age: 16, name: "John" },
        ]);
      });
      it('filter with "adult" return adult users', () => {
        expect(persons.adultFilter(users, "adult")).toEqual([
          { age: 19, name: "Alice" },
          { age: 21, name: "Jim" },
          { age: 32, name: "Kelly" },
        ]);
      });
    });
  }

  if (typeof persons.ageAverage === "function") {
    describe("ageAverage", () => {
      it("age average off all users is 20", () => {
        expect(persons.ageAverage(users)).toBe(20);
      });
      it("age average off all users is 14", () => {
        expect(persons.ageAverage(users, "child")).toBe(14);
      });
      it("age average off all users is 24", () => {
        expect(persons.ageAverage(users, "adult")).toBe(24);
      });
    });
  }

  if (typeof persons.isAllAdult === "function") {
    describe("isAllAdult", () => {
      it("users array with childs is not all adult", () => {
        expect(persons.isAllAdult(users)).toBe(false);
      });
      it("users array with only adult is all adult", () => {
        const adults = [
          { age: 19, name: "Alice" },
          { age: 32, name: "Kelly" },
        ];
        expect(persons.isAllAdult(adults)).toBe(true);
      });
    });
  }

  if (typeof persons.hasChild === "function") {
    describe("hasChild", () => {
      it("users array with childs has child", () => {
        expect(persons.hasChild(users)).toBe(true);
      });
      it("users array with only adult has no child", () => {
        const adults = [
          { age: 19, name: "Alice" },
          { age: 32, name: "Kelly" },
        ];
        expect(persons.hasChild(adults)).toBe(false);
      });
    });
  }

  if (typeof persons.addIsAdultProperty === "function") {
    describe("addIsAdultProperty", () => {
      beforeEach(() => {
        users = [
          { age: 12, name: "Bob" },
          { age: 21, name: "Jim" },
        ];
      });

      it("an isAdult property is added with correct value", () => {
        expect(persons.addIsAdultProperty(users)).toEqual([
          { age: 12, name: "Bob", isAdult: false },
          { age: 21, name: "Jim", isAdult: true },
        ]);
      });
      it("the returned array is a new array", () => {
        expect(persons.addIsAdultProperty(users)).not.toBe(users);
      });
      it("the returned array contains new objects", () => {
        expect(persons.addIsAdultProperty(users)[0]).not.toBe(users[0]);
        expect(persons.addIsAdultProperty(users)[1]).not.toBe(users[1]);
      });
    });
  }
});
```