**Задача 1**

**export interface User {**

**name: string;**

**age: number;**

**occupation: string;**

**}**

**export const users: User[] = [**

**{**

**name: 'Max Mustermann',**

**age: 25,**

**occupation: 'Chimney sweep'**

**},**

**{**

**name: 'Kate Müller',**

**age: 23,**

**occupation: 'Astronaut'**

**}**

**];**

**export function logPerson(user: User) {**

**console.log(` - ${user.name}, ${user.age}`);**

**}**

**console.log('Users:');**

**users.forEach(logPerson);**  
1) User описывает структуру объекта пользователя: name, age и occupation.

2)Массив users объявлен как массив объектов типа User.

3)Функция logPerson теперь принимает параметр типа User.

**Задача 2**

1)Для типа Teacher добавлен интерфейс WithLevel, чтобы включить уровень скила, и тип TeacherCourse, который включает роль учителя.

2)Тип TeacherInfo создан для использования в типе Director, он объединяет User, WithLevel и WithRate.

3)В типе Director обновлены словари students и teachers, чтобы они соответствовали необходимым структурам.

**type User = {**

**id: string;**

**name: string;**

**}**

**type Course = {**

**id: number;**

**title: string;**

**}**

**type WithRate = {**

**rate: 1 | 2 | 3 | 4 | 5;**

**}**

**type WithStudentRole = {**

**role: "student"**

**}**

**type WithTeacherRole = {**

**role: "teacher"**

**}**

**type WithLevel = {**

**level: "junior" | "middle" | "senior"**

**}**

**type StudentCourse = Course & WithStudentRole & WithRate & WithLevel**

**type Student = User & { courses: { [id: number]: StudentCourse } }**

**type TeacherCourse = Course & WithTeacherRole**

**type Teacher = User & WithLevel & { courses: { [id: number]: TeacherCourse } }**

**type TeacherInfo = User & WithLevel & WithRate**

**type Director = User & {**

**students: { [id: string]: User }**

**teachers: { [id: string]: TeacherInfo }**

**}**

**/\*-- Проверка --\*/**

**const s1: Student = {**

**id: "s1",**

**name: "s1",**

**courses: {**

**[1]: {**

**id: 1,**

**title: "First",**

**rate: 5,**

**role: "student",**

**level: "middle"**

**}**

**}**

**}**

**const t1: Teacher = {**

**id: "t1",**

**name: "t1",**

**level: "junior",**

**courses: {**

**[5]: {**

**id: 5,**

**title: "Fifth",**

**role: "teacher"**

**},**

**[1]: {**

**...s1.courses[1],**

**role: "teacher"**

**}**

**}**

**}**

**const d1: Director = {**

**id: "d1",**

**name: "d1",**

**students: {**

**["s1"]: s1,**

**["s2"]: {**

**id: "s2",**

**name: "s2"**

**}**

**},**

**teachers: {**

**["t1"]: {**

**id: "t1",**

**name: "t1",**

**level: "junior",**

**rate: 3**

**},**

**["t2"]: {**

**id: "t2",**

**name: "t2",**

**level: "senior",**

**rate: 5**

**}**

**}**

**}**

**Задача 3**

1. Добавлены generic-параметры <T, U>, чтобы можно было передавать массивы любых типов.
2. Указаны типы параметров first и second как T[] и U[].
3. Указан тип возвращаемого значения как Array<[T, U]>.

**function zip<T, U>(first: T[], second: U[]): Array<[T, U]> {**

**const minLength = Math.min(first.length, second.length);**

**const result: Array<[T, U]> = [];**

**for (let i = 0; i < minLength; i++) {**

**result.push([first[i], second[i]]);**

**}**

**return result;**

**}**

**const q1: Array<[number, string]> = zip([1, 2, 3, 4, 5, 6], ["1", "2", "3"]);**

**const q2: Array<[boolean, boolean]> = zip([true], [false, false]);**

**console.log(q1, q2);**

1. Добавлены generic-параметры **<T, K, V>**, где **T** - тип элементов в исходном массиве, **K** - тип ключей в результирующей карте, **V** - тип значений в результирующей карте.
2. Указаны типы параметров **source** как **T[]**, **keySelector** как **(item: T, index: number) => K**, **valueSelector** как **(item: T, index: number) => V**.
3. Указан тип возвращаемого значения как **Map<K, V[]>**

**function groupBy<T, K, V>(**

**source: T[],**

**keySelector: (item: T, index: number) => K,**

**valueSelector: (item: T, index: number) => V**

**): Map<K, V[]> {**

**const result = new Map<K, V[]>();**

**for (let i = 0; i < source.length; i++) {**

**const item = source[i];**

**const key = keySelector(item, i);**

**const value = valueSelector(item, i);**

**if (!result.has(key)) {**

**result.set(key, []);**

**}**

**result.get(key)!.push(value);**

**}**

**return result;**

**}**

**const q3: Map<number, number[]> = groupBy([1, 2, 3, 4], x => x % 2, x => x + 1);**

**const q4: Map<boolean, { x: string, i: number }[]> = groupBy(**

**["aaa", "bbb", "cc", "q", "lalaka"],**

**(\_, i) => i % 2 === 0,**

**(x, i) => ({ i, x })**

**);**

**console.log(q3, q4);**

**Задача 4**

1. Тип TeacherCourse:
   * Создан с использованием Omit<Course, "role"> и добавлением { role: Exclude<Role, "student"> }, чтобы типизировать курсы учителей без роли "student".
2. Тип Teacher:
   * Расширен User, добавлены поля level и courses, где courses является объектом с ключами типа number и значениями типа TeacherCourse.
3. Тип TeacherInfo:
   * Создан с использованием Pick<Teacher, "id" | "name" | "level"> и добавлением { rate: Rate }, чтобы включать рейтинг учителя.
4. Тип Director:
   * Расширен User, добавлены поля students и teachers, где students является объектом с ключами типа string и значениями типа Pick<User, "id" | "name">, а teachers является объектом с ключами типа string и значениями типа TeacherInfo.

**type User = {**

**id: string;**

**name: string;**

**}**

**type Role = "student" | "teacher"**

**type Rate = 1 | 2 | 3 | 4 | 5**

**type Level = "junior" | "middle" | "senior"**

**type Course = {**

**id: number;**

**title: string;**

**role: Role;**

**rate: Rate;**

**level: Level;**

**}**

**/\* --- \*/**

**type Student = User & { courses: { [id: number]: Omit<Course, "role"> & { role: Exclude<Role, "teacher">} } }**

**type TeacherCourse = Omit<Course, "role"> & { role: Exclude<Role, "student"> }**

**type Teacher = User & {**

**level: Level;**

**courses: { [id: number]: TeacherCourse };**

**}**

**type TeacherInfo = Pick<Teacher, "id" | "name" | "level"> & { rate: Rate }**

**type Director = User & {**

**students: { [id: string]: Pick<User, "id" | "name"> };**

**teachers: { [id: string]: TeacherInfo };**

**}**

**/\*-- Проверка --\*/**

**const s1: Student = {**

**id: "s1",**

**name: "s1",**

**courses: {**

**[1]: {**

**id: 1,**

**title: "First",**

**rate: 5,**

**role: "student",**

**level: "middle"**

**}**

**},**

**}**

**const t1: Teacher = {**

**id: "t1",**

**name: "t1",**

**level: "junior",**

**courses: {**

**[5]: {**

**id: 5,**

**title: "Fifth",**

**role: "teacher",**

**rate: 3,**

**level: "middle"**

**},**

**[1]: {**

**...s1.courses[1],**

**role: "teacher"**

**}**

**}**

**}**

**const d1: Director = {**

**id: "d1",**

**name: "d1",**

**students: {**

**["s1"]: {**

**id: "s1",**

**name: "s1"**

**},**

**["s2"]: {**

**id: "s2",**

**name: "s2"**

**}**

**},**

**teachers: {**

**["t1"]: {**

**id: "t1",**

**name: "t1",**

**level: "junior",**

**rate: 3**

**},**

**["t2"]: {**

**id: "t2",**

**name: "t2",**

**level: "senior",**

**rate: 5**

**}**

**}**

**}**