Ödev – 3

Aşama 1 Soruları: Bu sorularda, belirli bir console.log çıktısı alabilmek için sizden **çalışan bir kod (implementasyon)** yazmanız istenmektedir.

Aşama 2 Soruları: Bu sorularda, "ekran çıktısı" console.log değil, **TypeScript derleyicisinin verdiği hatadır**. Size hatalı bir kod verilecek. Amacınız, console.log kısmına dokunmadan, yalnızca **type veya interface tanımları yaparak** derleyicinin verdiği hataları (ekran çıktısını) düzeltmektir.

Aşama 1 Soruları: Runtime Çıktılı

Bu sorularda, `/* ??? */` ile işaretlenmiş yerleri doldurarak beklenen ekran çıktılarını almanız gerekmektedir. Ekran çıktılarına bağlantılar üzerinden erişebilirsiniz.

Soru 1: Jenerik Dizi Birleştirme `mergeArrays` fonksiyonunu, iki farklı tipte diziyi alıp tek bir dizide birleştirecek şekilde implemente edin.

Ekran çıktısı bağlantısı: https://pastecode.io/s/vx9sjmfx

Soru 2: Tip Koruyucuları (Type Guards) `Car` ve `Truck` sınıflarını ve `Vehicle` tipini kullanarak, `useVehicle` fonksiyonunu `if` bloğu içinde tip koruyucusu (`in` operatörü) kullanacak şekilde implemente edin.

Ekran çıktısı bağlantısı: https://pastecode.io/s/41ah3fyp

Soru 3: Sınıf ve Erişim Belirleyiciler `Logger` sınıfını implemente edin. `logHistory` dizisi `private` olmalı ve dışarıdan erişilememelidir.

Ekran çıktısı bağlantısı: https://pastecode.io/s/oc1wkc5o

Soru 4: `keyof` ile Jenerik Fonksiyon Bir nesne ve o nesnenin bir anahtarını alan `getProperty` fonksiyonunu implemente edin.

Ekran çıktısı bağlantısı: https://pastecode.io/s/5hpzfu4u

Soru 5: Fonksiyon Aşırı Yüklemesi (Overloading) Verilen `users` dizisini kullanarak, ID ile çağrıldığında tek bir kullanıcı, isimle çağrıldığında bir kullanıcı dizisi döndüren `search` fonksiyonunu implemente edin.

Ekran çıktısı bağlantısı: https://pastecode.io/s/cchk3ho1

Soru 6: Jenerik Sınıf Implementasyonu Bir `MemoryCache` sınıfı implemente edin. `set` ile veri eklemeli, `get` ile çekmeli ve `clear` ile temizlemelidir.

Ekran çıktısı bağlantısı: https://pastecode.io/s/axjez0bi

Soru 7: `Partial` ile Güncelleme Fonksiyonu `updateUser` fonksiyonunu, `Partial<User>` kullanarak mevcut bir kullanıcıyı güncelleyecek ve `Readonly<User>` olarak döndürecek şekilde implemente edin.

Ekran çıktısı bağlantısı: https://pastecode.io/s/gx38e658

Soru 8: Rest Parametreleri Aldığı tüm sayısal parametreleri toplayan `sum` fonksiyonunu "rest parameters" kullanarak implemente edin.

Ekran çıktısı bağlantısı: https://pastecode.io/s/7cwjbf4o

Soru 9: Soyut Sınıf (Abstract Class) `getArea` adında soyut bir metoda sahip `Shape` sınıfını ve ondan türeyen `Circle` sınıfını implemente edin. `Circle` alanı π×r2 olmalıdır.

Ekran çıktısı bağlantısı: https://pastecode.io/s/6brsokeo

Soru 10: Statik Özellikler ve Metotlar `MathHelper` sınıfına statik `PI` (3.14159) özelliğini ve statik `calculateCircumference` (Çevre = 2×π×r) metodunu ekleyin.

Ekran çıktısı bağlantısı: https://pastecode.io/s/dasgbrz2

Aşama 2 Soruları: Runtime Çıktılı

Bu sorularda, çalışan koda (`console.log` kısımlarına) **dokunmayacaksınız**. "Ekran çıktısı" derleyicinin verdiği hatadır. Amacınız, `/* ??? */` ile işaretlenmiş yerlerde **sadece Tip (type) tanımları** yaparak aşağıdaki kodun hatasız derlenmesini sağlamaktır.

Soru 11: Koşullu Tipler (Conditional Types) `UnwrapPromise<T>` tipini tanımlayın. Eğer T bir `Promise` ise içindeki tipi, değilse T'nin kendisini döndürmelidir.

İpucu: Burada tek satır içinde bir if-else kontrolü durumu var.

Ekran çıktısı: https://pastecode.io/s/5z7tbjxw

Soru 12: `infer` ile Fonksiyon Dönüş Tipi TypeScript'in `ReturnType<T>` tipini `GetReturnType<T>` olarak kendiniz yazın.

Ekran çıktısı: https://pastecode.io/s/4wmuiupj

Soru 13: Mapped Types ve Template Literals `CreateGetters<T>` tipini tanımlayın. `id: number` özelliğini `getId: () => number` metoduna dönüştürmelidir.

Ekran çıktısı: https://pastecode.io/s/n0nm1qyf

Soru 14: Rekürsif Tipler (Recursive Types) `Readonly<T>` sadece ilk seviyeyi salt okunur yapar. `DeepReadonly<T>` tipini, iç içe nesneleri de `readonly` yapacak şekilde tanımlayın.

Ekran çıktısı: https://pastecode.io/s/hvxwyixj

Soru 15: Mapped Types ile Filtreleme `PickByValueType<T, V>` tipini tanımlayın. Bir nesne (`T`) içinden, değeri (`V`) tipine uyan özellikleri seçmelidir.

Ekran çıktısı: https://pastecode.io/s/xs3vz9dy

Soru 16: Nominal Tipleme (Branded Types) `UserID` ve `PostID` tiplerini (Brand<T, U> kullanarak) tanımlayın. İkisi de `string` olmalı ama birbirine atanamamalıdır.

Ekran çıktısı: https://pastecode.io/s/58ktghr0

Soru 17: Dağıtılmış Koşullu Tipler (Distributive Conditionals) `FilterUnion<T, U>` tipini tanımlayın. Bir Union (`T`) içinden, `U`'ya atanabilen tipleri *çıkarmalıdır*.

Ekran çıktısı: https://pastecode.io/s/jovpm3um

Soru 18: `infer` ile Fonksiyon Parametresi `LastParameter<T>` tipini tanımlayın. Bir fonksiyonun *son* parametresinin tipini döndürmelidir.

Ekran çıktısı: https://pastecode.io/s/zs999o65

Soru 19: `infer` ile Dizi Elemanı Tipi `Flatten<T>` tipini tanımlayın. `T` bir dizi ise eleman tipini (`T[]` -> `T`), değilse `T`'yi döndürmelidir.

Ekran çıktısı: https://pastecode.io/s/00r20bmv

Soru 20: Template Literal Parsing (Rekürsif) Bu en zor sorulardan biridir. `ParseRouteParams<T>` tipini tanımlayın. `/users/:id` gibi bir string alıp `{ id: string }` gibi bir nesne tipine dönüştürmelidir.

Ekran çıktısı bağlantısı: https://pastecode.io/s/q4ktdgpv

Ödev Teslim:

Typescript Ödev şeklinde bir git reposu oluşturun Github hesabınız üzerinde
Github reponuzda soruları aşağıdaki şekildeki gibi ekleyin

- Soru1.ts
- Soru2.ts
- Soru3.ts
- ….
-

Repo public (herkese açık) olmalıdır.
Reponuzda bir adet Readme dosyası olması gerekmektedir. Readme dosyası içerisinde, repo hakkında bilgi verilecek, her soru için açıklamalar yapmalısınız.

Son Teslim tarihi: 7 Kasım 23:59

Not: Ödevlerinizin teslimini son dakikalara bırakmayınız