1- State whether each of the following is true or false. If false, explain why.

a) İstisnalar (Exceptions) her zaman ilk tespit edildikleri metot içinde işlenir.

- **Cevap: Yanlış (False).** Bir istisna, tespit edildiği metot içinde yakalanmazsa (catch edilmezse), çağrı yığını (call stack) boyunca yukarıya doğru iletilir. Eğer hiçbir yerde yakalanmazsa program çöker.

b) Kullanıcı tanımlı istisna sınıfları `SystemException` sınıfını genişletmelidir.

- **Cevap: Yanlış (False).** Microsoft'un güncel tavsiyesi, kullanıcı tanımlı istisnaların doğrudan `Exception` sınıfından türetilmesidir. `SystemException` genellikle işletim sistemi veya CLR tarafından fırlatılan hatalar için ayrılmıştır.

c) Dizi sınırları dışında bir indekse erişmek, CLR'ın bir istisna fırlatmasına neden olur.

- **Cevap: Doğru (True).** Bu durumda CLR, bir `IndexOutOfRangeException` fırlatır.

d) Karşılık gelen bir `catch` bloğu olmayan bir `try` bloğundan sonra `finally` bloğu isteğe bağlıdır.

- **Cevap: Yanlış (False).** Bir `try` bloğu kullanılıyorsa, arkasından ya en az bir `catch` bloğu gelmeli ya da bir `finally` bloğu gelmelidir. İkisi birden yoksa kod derlenmez.

e) Bir `finally` bloğunun çalışması garanti edilir.

- **Cevap: Doğru (True).** İstisna oluşsa da oluşmasa da (ve hatta `try` içinde `return` olsa bile) `finally` bloğu her zaman çalışır. Sadece bilgisayarın aniden kapanması veya `Environment.FailFast()` gibi uç durumlarda çalışmaz.

f) `return` anahtar kelimesini kullanarak bir istisnanın fırlatıldığı noktaya geri dönmek mümkündür.

- **Cevap: Yanlış (False).** C# "sonlandırma modeli" (termination model) kullanır. Bir istisna fırlatıldığında kontrol akışı o noktadan ayrılır ve geri dönülemez.

g) İstisnalar yeniden fırlatılabilir (rethrown).

- **Cevap: Doğru (True).** Bir `catch` bloğu içinde istisnayı yakalayıp bazı işlemler yaptıktan sonra `throw;` komutuyla hatayı üst katmanlara tekrar fırlatabilirsiniz.

h) `Exception` sınıfının `Message` özelliği, istisnanın fırlatıldığı metodu gösteren bir dize döndürür.

- **Cevap: Yanlış (False).** `Message` özelliği hatanın nedenini açıklayan bir metin döndürür. Hatanın hangi metottan geldiğini öğrenmek için `StackTrace` veya `TargetSite` özellikleri kullanılır.

i) Bir `when` ifadesi, bir `catch` bloğuna istisna filtresi ekler.

- **Cevap: Doğru (True).** `catch (Exception ex) when (ex.HResult == -2146233088)` gibi bir kullanım, hatanın sadece belirli şartlar altında yakalanmasını sağlar.

j) Boş bırakılabilir (nullable) bir değişkenin `Value` özelliği her zaman temel değeri döndürür.

- **Cevap: Yanlış (False).** Eğer değişken `null` ise `Value` özelliğine erişmeye çalışmak bir `InvalidOperationException` hatasına yol açar. Önce `HasValue` ile kontrol edilmelidir.

2- (Multiple Catch Blocks) Write a program to demonstrate that a single expression can throw different types of exceptions, which are caught in different catch blocks.

Çoklu Catch Bloğu Örneği

Bu programda kullanıcıdan bir sayı girmesini istiyoruz ve bu girdi üzerinden bölme işlemi yapıyoruz. Kullanıcının girebileceği farklı yanlış değerlere göre farklı hataları yakalayacağız.

```
using System;


class MultipleCatchTest
{
   static void Main()
   {
      try
      {
         Console.Write("Bir sayı giriniz: ");
         string input = Console.ReadLine();


         // Tek bir işlem satırı (veya blok) birden fazla hata türüne sebep olabilir:
         int sayi = int.Parse(input); // FormatException veya OverflowException fırlatabilir
         int sonuc = 100 / sayi;      // DivideByZeroException fırlatabilir
```

```
    Console.WriteLine($"Sonuç: {sonuc}");
}
// 1. Durum: Kullanıcı sayı yerine harf girerse
catch (FormatException)
{
    Console.WriteLine("Hata: Lütfen sadece rakam giriniz!");
}
// 2. Durum: Kullanıcı 0 girerse
catch (DivideByZeroException)
{
    Console.WriteLine("Hata: Bir sayı sıfıra bölünemez!");
}
// 3. Durum: Kullanıcı çok büyük bir sayı girerse (int sınırları dışı)
catch (OverflowException)
{
    Console.WriteLine("Hata: Girdiğiniz sayı çok büyük veya çok küçük!");
}
// 4. Durum: Beklenmedik diğer tüm hatalar için genel yakalayıcı
catch (Exception ex)
{
    Console.WriteLine($"Beklenmedik bir hata oluştu: {ex.Message}");
}

Console.WriteLine("Program akışı devam ediyor...");
```

```
    }

}
```