FTFa – Medlemsadministration

Baggrund:

Du skal udvikle et console-based medlemsadministrationsprogram i C#. Systemet skal kunne gemme medlemsoplysninger og beregne medlemmers dagpenge. Det er forventet at du bruger ca. 30-40 minutter på at løse det.

Formål:

Færdiggør medlemsadministrationsprogrammet ud fra den udleverede kode, og de nedenstående retningslinjer.

Udleveret kode:

Der medfølger 3 filer.

1. **Program.cs.** Her er skelettet for programmet, og det som vi vil bruge som vores ”frontend”.
2. **Database.cs.** Det er en mock af vores medlemsdatabase. Der er blevet tilføjet 3 eksempler af medlemmer, som du kan bruge til at teste og som kan være en del af databasen fra start, i den endelige løsning.
3. **Member.cs.** Er en class som repræsenterer et medlem i vores system. ToString er også blevet overridet, og den behøves ikke ændres.

Retningslinjer:

1. **Fokusområder:** Det primære mål er at se dine kodnings- og problemløsningsevner. Du skal ikke bruge tid på at lære hvordan dagpengeregler fungerer i Danmark, designe brugerfladen, eller lignende.
2. **BenefitAmount:** Den mængde dagpenge som et medlem skal have, er den mindste værdi af 90% af deres tidligere løn, og 19728. Altså Min(LastKnownSalary*0.9, 19728).
3. **Fleksibilitet:** Alt i de 3 udleverede filer kan ændres som du vil og er der bare for at hjælpe. De 3 filer kan helt fjernes/omdøbes/ændres og du bestemmer selv filstrukturen, så længe ”frontenden” stadig ser nogenlunde ens ud. Din måde at organisere og strukturere koden er en del af evalueringen.

Forventninger:

1. **Funktionalitet:** Implementér den funktionalitet, som er angivet i Program.cs.
    a. Registrering af nye medlemmer, og beregning af deres BenefitAmount.
    b. Visning af alle nuværende medlemmer.

    c. Søgning af medlemmer, ud fra deres FullName og/eller deres Email.
    d. Opdatering af IsEmployed for et medlem der skifter employment status, ud fra deres ID.
    e. Sletning af et medlem ud fra deres ID.
2. **Validering:** Det er ikke forventet at du finder et perfekt regex til at validere emails, eller tjekker om navnene er gyldige. Kun tænk over simpel validering, som tomme emails/navne og negativ LastKnownSalary. Ikke brug tid på at tænke på forskellige edge-cases.
3. **Kode kvalitet:** Din kode skal være effektiv, organiseret og nem at vedligeholde. Brug comments de steder hvor du laver mere komplekse eller uklare ting.
4. **Test:** Sørg for selv at teste at funktionaliteten virker, men brug ikke tid på at skrive unit tests.

Aflevering:

Når du har afleveret casen, vil vi gerne have dig til at gennemgå din løsning med os. Vi er nysgerrige efter ting som:

1. Dine design beslutninger.
2. De udfordringer du støtte på, og hvordan du løste dem.
3. De antagelser du har lavet.
4. Hvordan du kunne forbedre løsningen, havde du haft mere tid.

Vi er lige så interesserede i at høre din tankegang igennem processen som din endelige løsning.