Tre projektuppgifter

1. Bergvärme – simulering av borrutrustning

En borrutrustning för bergvärme kan borra 25 m under den 1:a timmen i en viss tomtmark.

Under de följande timmarna minskar borrens prestation med uppskattningsvis 10-20% per timme. Den exakta minskningen är inte känd, då den är beroende av markförhållandena. Borren ska gå oavbrutet i 8 timmar.

Skriv ett JavaScript program som uppskattar det totala borrdjupet.

Ledning: Börja med att simulera minskningen av borrens prestation efter den 1:a timmen med slumptal mellan 10 och 20. Summera borrhålets djup efter den 1:a timmen baserad på denna simulation.

Skriv ut slutligen ett närmevärde för borrhålets totala djup efter 8 timmar.

Skriv ut även borrprestationens procentuella minskning per timme vid den aktuella körningen, t.ex.:

”Denna uppskattning baseras på 12% minskning av borrprestationen per timme.”

P.g.a. simuleringen med slumptal borde man få vid olika körningar olika procentsatser för minskningen av borrens prestation och därmed även andra uppskattnningar av det totala borrdjupet. I praktiken duger dock ofta en sådan uppsattning, då den kan vara en värdefull information för planeringen av arbetet.

En körning kan t.ex. se ut så här:

2. Palindrom – en lek med ord

En *palindrom* är en sträng som inte ändras när den läses baklänges. T.ex. är orden *rar*, *död* och *radar* palindromer, även namnet *Hannah* när det stavas så. Men även en text som *ni talar bra latin* är en palindrom om man ignorerar mellanslagen. Och det ska man göra. Därför: vid behandling av sådana texter i ett program låt koden först ta bort alla mellanslag.

Skriv en funktion **`palindrom()`** som avgör om en sträng är en *palindrom* eller ej. Anropa sedan funktionen i ett JavaScript program som hanterar strängar. Låt användaren mata in strängar – med eller utan mellanslag – så länge tills man hittat en palindrom. Bjud på möjligheten att avsluta om ingen palindrom hittas och föreslå användaren ett antal palindromer.

3. Frekvenstabell – stämmer sannolikhetsläran?

Skriv ett JavaScript program som simulerar tärningskast, dvs genererar slumptal mellan **`1`** och **`6`**, och ställer upp en frekvenstabell enligt följande beskrivning:

Frekvens är antalet förekomster av ett resultat (utfall) bland tärningens 6 möjliga.

Låt programmet genomföra olika antal simuleringar och räkna vid varje simulering frekvensen för varje resultat **`1`**, ...,**`6`** av tärningskastet. T.ex. ska man kunna läsa av från tabellen hur många gånger resultatet **`1`** förekommer när man kastar tärningen 50 gånger, 100 gånger, 1 000 gånger, 5 000 gånger, 10 000 gånger, osv. Avgör själv hur långt du går. Samma information ska man kunna läsa av från tabellen om tärningskastets andra resultat **`2`**, ... ,**`6`**.

Infoga i tabellen även en kolumn som för varje resultat av tärningskastet visar kvoten:

Frekvens / Antalet tärningskast

Denna kvot är den experimentella sannolikheten för ett visst resultat. Undersök på vilket sätt den experimentella sannolikheten närmar sig den ideala sannolikheten för varje resultat, som enligt sannolikhetsläran borde vara 1/6 eller 0,16667.