Välkomna alla SUVx-23-are till ***Boiler room!***

Följande uppdrag ska genomföras under Boiler roomen ***fre 15/12, kl 9-12*** och ***fre 12/1, kl 9-12.***

Om genomförandet

Eftersom Boiler room kommer att genomföras varje vecka fr.o.m. vt 2024, istället för varannan vecka, med uppdrag som sträcker sig över en längre tid, vill vi i detta uppdrag öva oss på att genomföra uppdrag som går över *två* Boiler room tillfällen: **fre 15/12** och **fre 12/1**. Som ni vet, förskjöts det Boiler room som var planerat fre 29/12 till fre 12/1. Så, dela in tiden så att ni hinner att genomföra följande fyra uppgifter och lämna in dem som *ett* uppdrag **fredagen den 12/1-2024**. Ni kan själva välja vilken eller vilka uppgifter ni prioriterar tidsmässigt. Därför: läs igenom först alla uppgifter.

Konsekvensen blir att även tiden mellan dessa två tillfällen kan användas för att arbeta med uppdraget. Men även detta kan betraktas som en övning på det nya konceptet av Boiler room som skolan kommer att realisera efter nyår.

1. 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 **`bool palindrom(char *a)`** som avgör om en sträng är en *palindrom* eller ej. Anropa sedan funktionen i **`main()`** efter inmatning av en sträng som ska testas, i ett C++ program som hanterar strängar med pekare. 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.

2. Bergvärme – det omvända problemet

I ett tidigare Boiler room uppdrag behandlades problemet med att uppskatta borrdjupet som en borrutrustning för bergvärme kunde åstadkomma när borren gick i 8 timmar. Så här beskrevs uppdraget:

> 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 program som uppskattar det totala borrdjupet.
>
> 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 även ut borrprestationens procentuella minskning per timme vid den aktuella körningen, t.ex.:
>
> *”Detta närmevärde baseras på en 14%-ig minskning av borrens prestationen per timme efter den första timmen.”*

Det nya uppdraget behandlar det omvända problemet:

Skriv ett program som läser in ett önskat totalt borrdjup och beräknar samt skriver ut antalet arbetstimmar som borren uppskattningsvis skulle behöva för att nå detta borrdjup.

3. Frekvenstabell – stämmer sannolikhetsläran?

Skriv utgående från programmet **`Dice`** (**kursboken**, sid 161) som simulerar tärningskast, ett program som 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.

4. Fibonacci – en rekursiv talsekvens

År 1202 formulerade den italienske matematikern *Leonardo Pisano Fibonacci* i sin bok *Liber abaci* (Boken om räknekonsten) följande uppgift som handlar om kaniners fortplantning. Han observerade följande:

> Ett kaninpar föder från den andra månaden av sin tillvaro ett nytt par varje månad. Samma gäller för de nya paren.
>
> Hur många par kommer att finnas om två år, om Fibonaccis observation är korrekt?

Skriv ett program som besvarar frågan ovan. Programmet ska vara generellt, så att det även ger svar på: Hur många kaninpar kommer att finnas om **n** månader.

I programmering är lösningen av Fibonaccis problem (ovan) känd som ett typexempel för s.k. *rekursiva* algoritmer.

Rekursion är ett koncept inom problemlösning och programmering som tillämpar successiv upprepning av vissa beräkningar. Rekursiva metoder och funktioner är sådana som anropar sig själva, ungefär som hundar som bitar sig i svansen. Ordet rekursiv kommer från *recurrere* på latin som på engelska betyder *to run back* eller *to run again* dvs att gå tillbaka och köra igen.

Fibonaccis problem ger upphov till en rekursiv talsekvens, de s.k. *Fibonaccitalen*. Dvs varje tal i denna sekvens kan beräknas genom en kombination av tal som förekommer i sekvensen *före* det aktuella talet och är därför redan kända. Genom att upptäcka ett mönster mellan **n** och $\mathbf{F_n}$ kan en formel ställas upp, kallad *Fibonaccis rekursionsformel*, där:

n = Antalet månader
F_n = Antalet kaninpar i månaden n

Försök att ställa upp denna formel och implementera den i en funktion. Dvs skriv en funktion **`int Fib(int n)`** som beräknar Fibonaccitalen och anropa den i **`main()`** för **`n = 1, 2, ... 30`**.

Lycka till!

Hälsningar

Taifun