MA1508

Inlämningsuppgifter 1 *Seyed Mostafa Mohseni*

1. **Kontrollera att ditt eget personnummer är korrekt med Luhn-algoritmen.**

Kontroll av mitt personnummer (010722-4495).
2.0= 0, 1.1=1, 2.0=0, 1.7=7, 2.2=4, 1.2 = 2, 2.4= 8, 1.4 =4, 2.9=18-9=9, 1.5 =5.
Summa = 0+1+0+7+4+2+8+4+9+5 = 40. Delbart med 10!
40 mod 10 = 0 inga fel.

2. **Ändra en siffra i ditt personnummer och visa att Luhn-algoritmen säger att det blir fel.**

0107224395. Vi ändrar 4 till 3.
2.0= 0, 1.1=1, 2.0=0, 1.7=7, 2.2=4, 1.2 = 2, 2.4= 8, 1.3 =3, 2.9=18-9=9, 1.5 =5.
Summa = 0+1+0+7+4+2+8+3+9+5 = 39.
39 mod 10 = 9 fel personnummer.

3. **Byt plats på två grannsiffror i ditt personnummer (som inte är 0 och 9) och visa att Luhn-algoritmen säger att det blir fel.**

Jag väljer de två grannarna 4 och 9 (position 8 och 9).
Nytt nummer: 0107224945
2.0= 0, 1.1=1, 2.0=0, 1.7=7, 2.2=4, 1.2 = 2, 2.4= 8, 1.9 =9, 2.4=8, 1.5 =5.
Summa = 0+1+0+7+4+2+8+9+8+5 = 44
44 mod 10 = 4 != 0 fel personnummer.

4. **Ändra två siffror i ditt personnummer på ett sådant sätt att Luhn-algoritmen inte upptäcker att något blivit fel.**

Efter några tester så hittade ett alternativ, jag ändrar de två första siffrorna 01 → 19
nytt nummer: 1907224495
2·1 = 2, 1·9 = 9, 2·0 = 0, 1·7 = 7 , 2·2 = 4, 1·2 = 2, 2·4 = 8, 1·4 = 4, 2·9 = 18−9 = 9, 1·5 = 5
Summa = 2 + 9 + 0 + 7 + 4 + 2 + 8 + 4 + 9 + 5 = 50
50 mod 10 = 0 giltig personnummer.

5. **Förklara varför vi är speciellt intresserade av att upptäcka fel där två siffror byter plats när vi jobbar med personnummer, men inte har något sådant speciellt intresse när Alice och Bob kommunicerar över en kabel.**

Vi vill upptäcka grannbyten i personnummer eftersom det är ett vanligt mänskligt skrivfel. Vid kabelöverföring är felen i slumpmässiga bitfel, så där behövs inte samma skydd mot just transpositionsfel.

6. **Lämna in Python-kod för en rutin som gör kontrollen av ett personnummer med Luhns algoritm. Du kan anta personnumret är representerat som en sträng eller en integer beroende på vad du tycker är enklast.**

```
def  luhna_check(p_number: str) -> bool:
  # Tar bort bindestreck om det finns
  p_number = p_number.replace("-", "")

  # Måste vara exakt 10 siffror
  if len(p_number) != 10 or not p_number.isdigit():
     return False

  total_sum = 0
  for i, char in enumerate(p_number):
     num = int(char)
     if i % 2 == 0:
        num *= 2
        if num > 9:
           num -= 9
     total_sum += num

# returnera True om summan är delbar med 10
  return total_sum % 10 == 0
```

7. **Ett alternativ till Luhns-algoritm hade varit att använda samma algoritm som för ISBN-nummer. Men då hade sista siffran i personnumret kunnat behöva vara ett X (som står för 10) Räkna ut vad din kontrollsiffra hade varit med ISBN-algoritmen.**

0107224495

Multiplicerar och summerar produkterna av de första 9 siffrorna med vikterna 10-2:

$(0·10)+(1·9)+(0·8)+(7·7)+(2·6)+(2·5)+(4·4)+(4·3)+(9·2)$

$= 0 + 9 + 0 + 49 + 12 + 10 + 16 + 12 + 18 = 126$

Räknar kontrollsiffran:

Hitta d10 så att : **$126 + d10 \equiv 0 \pmod{11}$**

126 mod11 = $11·11=121$ och $126−121=$ <u>5</u> alltså om vi delar 126 på 11 blir resten 5!

För att summan ska bli jämnt delbar med 11, måste vi lägga till något tal d10 som gör att resten blir 0 . Alltså: $5 + d10 \equiv 0 \pmod{11}$

Om d10=6 :

$5+6 =11\equiv 0 \pmod{11}$ därför är kontrollsiffran d10= **<u>6</u>**