Tentamen
EDAA05 Datorer i system
Lösningsförslag

2011–10–17, 08.00–13.00

---

1. a) 100011010 och 11A
   b) 10011101
   c) 62
   d) 111101000011011
   e)
```
  1111
  --------
   1011010
+   101001
---------
  10000011
```
   f)
```
   xxxxxx
   1100100
-   100101
---------
    111111
```

2. a) 1) De första tre byten i filen utgör en så kallad BOM (Byte Order Mark). BOM förekommer när någon teckenkodningsstandard som bygger på Unicode används, t.ex. UTF-8. En BOM anger exakt vilken Unicode-variant som använts, UTF-8, UTF-16 eller UTF-32. Den anger också i förekommande fall i vilken ordning byten som bygger upp ett tecken kommer (big-endian eller little-endian).
      2) Svenska tecken verkar ha kodats som två bytes. Detta är karaktäristiskt för UTF-8. Om man matchar de två byten mot hur UTF-8 kodar svenska tecken ser man en överensstämmelse.
   b) Teckenkod 10, ”\n”, newline eller linefeed.
   c) Unicode är en förteckning över hur olika tecken motsvaras av en teckenkod. Den säger dock ingenting om hur denna teckenkod exakt lagras i datorns minne. UTF-8 är ett exempel på en standard som talar om hur ett utvalt tecken i Unicode-tabellen ska lagras i form av en sekvens av bytes (eller ettor och nollor).

3. a) Sanningstabellerna blir för lag 1...

p	q	$\neg(p \wedge q)$	$\neg p \vee \neg q$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

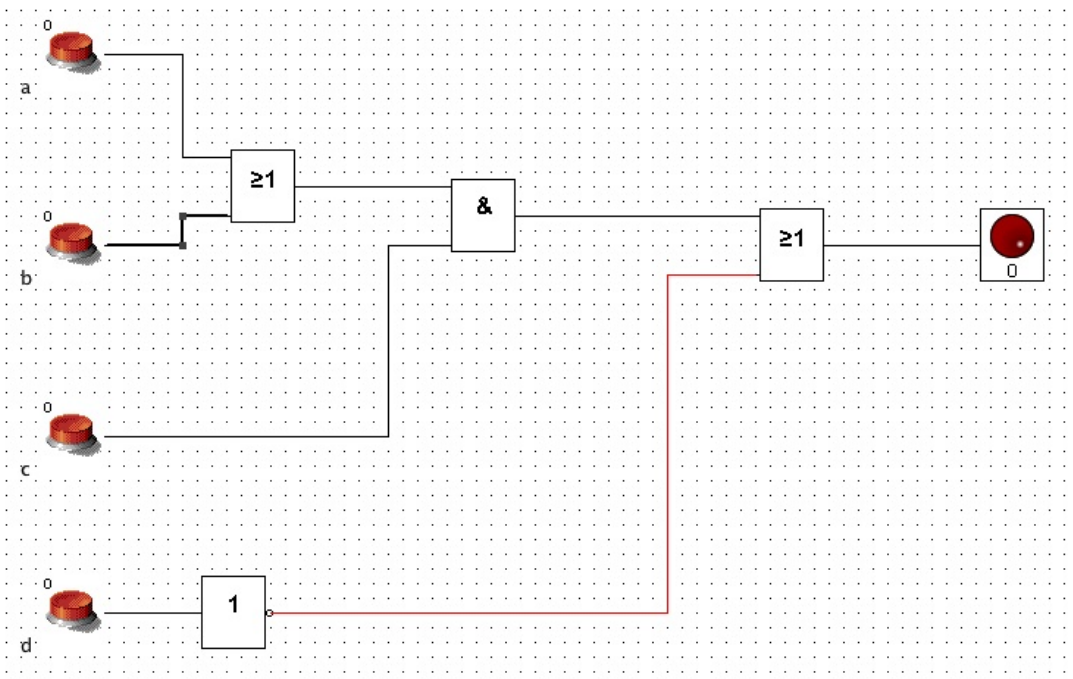
samt för lag 2...

p	q	$\neg(p \vee q)$	$\neg p \wedge \neg q$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

b) Uttryck 1:
$(p \wedge \neg q) \vee (\neg p \wedge \neg q) = (p \vee \neg p) \wedge \neg q = 1 \wedge \neg q = \neg q$
Uttryck 2:
$p \wedge \neg q \vee \neg(r \vee \neg q) \vee r \wedge q = p \wedge \neg q \vee \neg r \wedge q \vee r \wedge q = p \wedge \neg q \vee q \wedge (\neg r \vee r) =$
$p \wedge \neg q \vee q = (p \vee q) \wedge (\neg q \vee q) = p \vee q$

4. a) $x = \neg a \wedge (b \vee \neg c)$

b) Uttrycket kan t.ex. realiseras på följande sätt (ritat i LogicSim):



5. a) $o = \neg s1 \wedge \neg s2 \wedge i \vee \neg s1 \wedge s2 \wedge \neg i \vee \neg s1 \wedge s2 \wedge i$
   $s1' = \neg s1 \wedge \neg s2 \wedge i \vee \neg s1 \wedge s2 \wedge i$
   $s2' = s1 \wedge \neg s2 \wedge i$

   b) Den färdiga maskinen kan realiseras på följande (ritat i LogicSim):