EI Info 11 2010-11	*INFORMATIK* 1. Klausur	

Unsere erste Informatik-Klausur! Du kannst deinen GTR verwenden. Achte auf eine übersichtliche und gut erläuterte Darstellung! **(Bearbeitungszeit: 60 Minuten)**

1. Aufgabe **(7 Punkte)**

a) Was ist ein Bit? Was ist ein Byte? Was ist ein Megabyte?

Ein Bit ist die kleinste Informationseinheit in der Informatik. Beispiele: Strom an bzw. Strom aus, An – Aus, 0 – 1 usw. Ein Byte entspricht 8 Bits. Mega steht für Millionen, also ist 1 Megabyte = 1.000.000 Byte (wobei eigentlich nicht 1000, sondern 1024 in der Informatik dem Kilo und 1024^2 dem Mega entspricht).

b) Übersetze folgende Binär-Bytes ins Dezimalsystem und ins Hexadezimalsystem:

$0110\ 0100_2$, $1010\ 0001_2$, $0011\ 1111_2$.

1. Zahl: 100, 64_{16}, 2. Zahl: 161, $A1_{16}$, 3. Zahl: 63, $3F_{16}$.

c) Stelle die folgenden Hexadezimalsystem-Zahlen im Dezimalsystem dar:

10_{16}, $9F_{16}$, $F9_{16}$.

1. Zahl: 16, 2. Zahl: 159, 3. Zahl: 249.

d) Stelle die Zehnersystemzahl 69 im Hexadezimalsystem und als Byte im Binärcode dar.

Hex: 45, Byte: 01000101.

e) Wieviele verschiedene Zustände lassen sich mit einem Binär-Byte beschreiben?

Insgesamt 256 verschieden Zustände.

2. Aufgabe **(4 Punkte)**

Du bekommst folgenden Code für ein Schwarzweiß-Bild geliefert:

00000110000001101000010100100011010100100100001

In diesem Code ist im ersten Byte die Breite des Bildes in Pixeln definiert, im zweiten Byte die Höhe in Pixeln. Ansonsten stellt die Zahl 1 die Farbe Schwarz dar und 0 die Farbe Weiß.

a) Bestimme die Bildgröße.

Das erste Byte wäre hier 00000110 **und das entspricht einer Breite von 6 Pixeln. Das zweite Byte** 00000110 **entspricht einer Höhe von 6.**

b) Zeichne das Bild. Stelle jeden Pixel mit einem Kästchen dar.

100001
010010
001101
010010
100001

Die letzte Zeile fehlt (hatte ich ganz einfach vergessen…) und damit werden hier die Werte entweder auf default=0 stehen oder es steht irgendetwas im Speicher!

3. Aufgabe **(3 Punkte)**

Vereinfache folgende boolesche Terme so weit wie möglich. Forme dabei übersichtlich um!

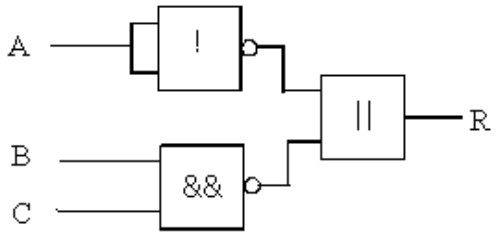
a) $A \wedge (\neg A \vee B)$

b) $(B \vee C) \wedge (A \vee B) \wedge (B \vee \neg B)$

In a) ist der Schnitt von A mit nichtA sicher leer. Daher bleibt nur A oder B!
In b) ist in der hinteren Klammer B oder nicht B immer wahr. Ist B wahr, stimmen die beiden anderen Klammern. Ist B falsch, dann müssen A UND C wahr sein, sonst wird's wegen dem Schnitt der beiden Klammern nichts mehr. Also ist die Lösung „B oder (A und C)".

4. Aufgabe **(2 Punkte)**

In dem Schaltnetz unten gibt es drei Eingänge A, B und C und einen Ausgang R.



a) Welchen Wert hat R, wenn A=1, B=1 und C=0 gilt?

Für A: nicht 1 ist 0. Für B und C wird durch UND eine 0, weil C=0 ist. Jetzt gilt 0 ODER 0, also R=0.

b) Notiere für R einen passenden Booleschen Term.

R = !A ODER (B UND C).

5. Aufgabe **(2 Punkte)**

Bestimme nachvollziehbar (!) den Wahrheitswert (w/f) folgender Aussage:

$$((-3 < -4) \vee (-5 \geq 4)) \wedge (\neg(16 + 27 = 38))$$

-3 ist größer als -4 (muss man aufpassen…) und -5 ist kleiner als 4. Damit ist die Aussage sicher falsch, auch wenn die dritte Klammer wahr ist wegen der Verneinung von etwas Falschem.

6. Aufgabe **(4 Punkte)**

Hier eine act-Methode von Robita aus dem bekannten Roboterszenario:

```
public void act()
{
	if (wandVorne())
	{
		dreheRechts();

		if (akkuAufFeld() && wandVorne())
		{
			akkuAufnehmen();
		}
		else
		{
			bewegen();
		}
	}
}
```

Beschreibe detailliert, welcher Vorgang abläuft, wenn du hier die play-Taste drückst:

Da keine Wand vorne ist, macht Robita gar nichts!

7. Aufgabe **(8 Punkte)**

Robita beginnt in einer Position, in der eine Mauer rechts von ihr ist. Sie soll entlang der Mauer gehen und im Falle einer Mauerlücke auf die andere Seite der Mauer wechseln und dort weitergehen. Die Abfolge der Mauerlücken ist dabei unbekannt und es gibt nur eine Lückenmauer pro Szenario. Ein Beispiel:

Schreibe für Robita eine act-Methode, die bewirkt, dass sie durch die Mauerlücken „Slalom" läuft. Nutze Java- || Pseudo-Code und verwende für Robitas Aktionen ausschließlich folgende Methoden: **wandRechts()**, **wandLinks()**, **bewegen()**, **dreheRechts()**.

Hier nur der Ansatz; es gibt entweder eine Wand rechts, dann wird bewegt. Else geht's nach unten. Innerhalb dieser Else-Bedingung findet sich wieder eine if-else-

Bedingung; solange keine Wand links kommt, geht es geradeaus weiter. Kommt eine Lücke, geht's wieder nach oben. Danach endet die ELSE-Bedingung von wandRechts erst! Jetzt ist Robita wieder oben und läuft weiter.