UIT2201: Computer Science and Information Technology Revolution
Fall Semester 2014 – Finals (Solution Sketch)

(NOT TO BE GIVEN TO FUTURE UIT2201 STUDENTS)

Fun Question: MatheMAGIC!

Question 1: (a)--(f) T F T F F T

(g) Closed-world assumption in KBS mean that knowledge-base makes true/false decisions based only on the facts given to the system (it is a closed-world). In particular, what is not known to be true, is assumed to be false.

Question 2: (15 points)

(a)(i) SQL Query:

```
SELECT  SI.S-ID, Name, Faculty, Course-ID
FROM    SI, EN
WHERE  (SI.Faculty = "FASS")
   AND (EN.Course-ID = "CS3230")
   AND (SI.S-ID = EN.S-ID);
```

(a)(ii) Using Basic Primitives:

```
A1  ← e-select FROM SI WHERE (Faculty = "FASS");
A2  ← e-select FROM EN WHERE (Course-ID="CS3230");
A3  ← e-join A1 and A2 WHERE (A1.S-ID = A2.S-ID);
Ans ← e-project SI.S-ID, Name, Faculty, C-ID FROM A3;
```

(b)(i) H[1..5] = [1, 0, 2, 3, 2]

(b)(ii) Fancy algorithm: **(b)(iii)** $\Theta(n)$

Step 1: Set all H[k]=0 for k=1,2,3,4,5
Step 2: Loop through BS[j], j=1,2,…,*n*
H[BS[j]] := H[BS[j]] + 1

Another method:

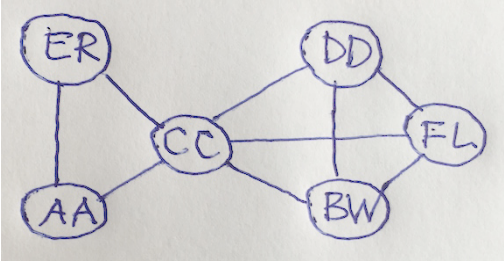
Re-Use the algorithm that counts the occurrences of X in array.
Loop through k=1,2,3,4,5
H[k] ← Count-Occ (*BS*, *n*, *k*) [Done in tutorials]
This is also $\Theta(n)$ running time.

There are other methods that you can also explore.

Question 3: (15 points)

(a) In the conflict graph, each node (vertice) represents a *committee*. There is a conflict edge between two vertices (committees) *u* and *v*, if there is a faculty member who is in both committee *u* and *v*.
Then we colour the vertices in the conflict graph *G* so that adjacent vertices have different colours. And we want to *minimize* the number of colours. The vertices of the same colour do not conflict and so the corresponding committee meetings can be held together. Hence, by minimizing the number of colours, we minimize the number of time slots for the meetings.

(b) The conflict graph G=(V,E)



(c) The Schedule:

4 Colours are used; CC=1, DD=2, BW=3, FL=4, ER=2, AA=3;

Time-Slot-1: CC	Time-Slot-2: DD, ER
Time-Slot-3: AA, BW	Time-Slot-2: FL

(d) Transformation: DIY (book work / lecture notes)

Question 4: (15 points)

(a) Truth table for F = A*~B + ~A*C + B*C

A	B	C	F	A*~B	~A*C	B*C
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	1	1	0	1	1
1	0	0	1	1	0	0
1	0	1	1	1	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	1

```
==========================================
Optional Alternative Method:
==========================================
F = 001 + 011 + 100 + 101 + 111
  = (001+011) + (100+101) + (101+111)
  = 0X1 + 10X + 1X1
  = (0X1+1X1) + 10X = XX1 + 10X = C + A(~B)
==========================================
```

(b) F = (~A)(~B)C + (~A)BC + A(~B)(~C) + A(~B)C + ABC

(c) F = (~A)C [~B+B] + A(~B) [C + ~C] + AC [~B + B]
= (~A)C + A(~B) + AC = C [~A + A] + A(~B)
= C + A(~B)

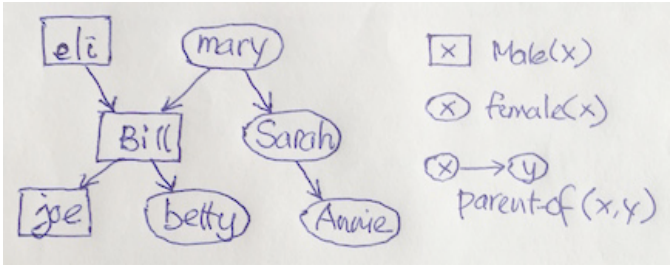
(d) Logic Circuit -- DIY

(e) The set {*, +, ~} is *logically complete* means that any logical function can be implemented using some combination of *, + and ~ gates.

(f) We know that the set {*, +, ~} is *logically complete.*
By de Morgan's law, A*B = ~(~(A*B)) = ~(~A+~B).
And so, in any circuits with {*,+,~}, we just replace any * by + and ~ as show above.

Question 5: (15 points)

(a)(i) Relationship Diagram:



SSS

(a)(ii)

```
R1: father-of(X,Y)    if parent-of(X,Y), male(X)
R2: ancestor-of(X,Y)  if parent-of(X,Y)
R3: ancestor-of(X,Y)  if parent-of(X,Z), ancestor-of(Z,X)
```

(a)(iii)

```
-? parent-of(P, sarah)      Answer:  P = mary
-? parent-of(Q, bill)       Answer:  Q = eli
-? ancestor-of(S,annie)     Answer:  S = sarah, mary
```

(b) [See solution to T12-Q2]

~~~ END OF QUIZ ~~~