1 Graphs, Circuits and Positional Number Systems

Problem 1.1 (Planar Graphs)

A graph G is called planar if G can be drawn in the plane in such a manner that edges do not cross elsewhere than vertices. The geometric realization of a planar graph gives rise to regions in the plane called faces; if G is a finite planar graph, there will be one unbounded (i.e. infinite) face, and all other faces (if there are any) will be bounded. Given a planar realization of the graph G, let $v = \#(V)$, $e = \#(E)$, and let f be the number of faces (including the unbounded face) of G's realization. 15pt

Prove or refute the Euler formula, i.e. that $v - e + f = 2$, must hold for a connected planar graph.

Solution:

Proof: Proof by induction on the number of faces

P.1 If G has only one face, it is acyclic and connected, so it is a tree and $e = v - 1$. Thus $v - e + f = 2$.

P.2 Otherwise, choose an edge e connecting two different faces of G, and remove it; e can then only appear once in the boundary of each face, so the graph remains connected – any path involving e can be replaced by a path around the other side of one of the two faces. This removal decreases both the number of faces and edges by one, and the result then holds by induction. □

Problem 1.2 (Designing a Logic Circuit)

Given the following truth table: 15pt

x	y	z	OutputA	OutputB
0	0	0	1	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

1. Obtain and simplify (using any method) logic expressions for OutputA and OutputB.
2. Implement the corresponding logic circuit using only AND, OR, NOT gates, and using the least number of these.
3. Implement the same circuit (same logic expression) using the least number of NAND gates.

Note: For the last part, you do not have to prove that the number of gates you used is minimal. However the number of gates should be smaller than in the circuit obtained just by directly expressing each OR/AND/NOT gate with NAND gates.

Solution:

Problem 1.3 (Years and bases) 5pt

Explain what are the smallest and largest numbers that can be computed in each of the following bases: unsigned binary on 10 bits, hexadecimal on 3 characters, two's complement in 5 bits (state the value in decimal and the respective system and briefly say why).

Then convert $x = DDMM$ (where DD represents the day and MM the month of this tutorial) into binary in 12 bits, hexadecimal in 3 characters, TCN in the minimum number of bits (state which one it is).

Solution:

Binary in 12 bits: $[0, 1023]$
Hexadecimal in 3 characters: $[0, 4095]$
TCN in 5 bits: $[-16, 15]$

$x = 2305$ which can be written as:

Binary 12-bits: 100100000001
Hexadecimal 3-char: 901
TCN minimum number of bits: 0100100000001 , where the minimum number of bits is 13.

2 Machine Languages

Problem 2.1: Write a $\mathcal{L}(\texttt{VMP})$ program that takes 9 arguments, each one corresponding to a unique place in a 3x3 matrix, and calculates the determinant of said matrix. 15pt
E.g. if the arguments are 1,2,3,4,5,6,7,8,9, they represent the following matrix:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Hint: The determinant of a 3x3 matrix can be calculated by executing the following steps:

- Add the product of the entries on left-right diagonal of length 3(wrap around if necessary) starting at each of the elements in the first row
- Subtract the product of the entries on right-left diagonal of length 3(wrap around if necessary) starting at each of the elements in the first row
- You can also use any other method that you find easier, this is the classic approach to finding a 3x3 matrix determinant

Problem 2.2 (Formal languages with Turing Machines)

20pt

Design and implement a Turing Machine that, given a string surrounded by #, determines whether it is of the form $0^n1^{2n}0^n$. It should end in the state "yes" if the string matches the pattern and in the state "no" otherwise. You can assume that $n > 0$ and that the head will be positioned at the first non-# character of the input. Please add comments on how the general idea works.

Solution:

```
0 # # r s0

s0 0 s r s1
s0 1 1 r no
s0 s s r s0
s0 # # r yes

s1 0 0 r s1
s1 1 s r s2
s1 s s r s1
s1 # # r no

s2 0 0 r s2
s2 1 1 r s2
s2 s s r s2
s2 # # l s3

s3 0 s l s4
s3 1 1 l no
s3 s s l s3
s3 # # r no

s4 0 0 l s4
s4 1 s l s5
s4 s s l s4
s4 # # r no

s5 0 0 l s5
s5 s s l s5
s5 1 1 l s5
s5 # # r s0
```

3 Internet & WWW

Problem 3.1 (Internet Protocol Suite)

What is the *Internet Protocol Suite*? Define its structure as detailed as possible. 5pt

Solution: The Internet Protocol Suite is the set of communications protocols used for the Internet and other similar networks. It is structured into four layers, namely

- the Application Layer
- the Transport Layer
- the Internet Layer
- the Link Layer

An application uses a set of protocols to send its data down the layers, being further encapsulated at each level.

Problem 3.2 (Internet Overview)

Please explain with as much detail as you can: 10pt

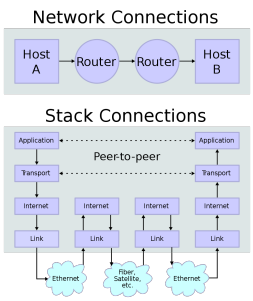
1. Through what layers does a packet go when traveling from one host to another? Start with application layer from host A and finish with application layer in host B. Provide a graph and give a detailed explanation of what happens in each layer.
2. What are TCP and UDP? What is the difference between the two? List at least three differences.
3. What is an URI? Explain what is the syntax of an URI. You can use the following example to help you in your explanation:

   `http://thelast.net:80/problem?course=gencs2#solution`

4. What is TLS? Specify the steps which are executed by a client and a server in a TLS handshake.

Solution:

1. The packet goes down from the application layer on A to the link layer. Then on each routing hop it goes up to the Internet link, where based on the destination IP address it is decided where to route it. At the destination B it goes up from the link layer to the application layer. Graph can be seen on slide 340 in the notes:



2. TCP Transmission Control Protocol
   UDP User Datagram Protocol
   TCP is manly used in the Internet layer, UDP in the transport layer.
   TCP is used for commutation. UDP for multicasting and broadcasting.
   TCP is connection oriented, UDP is connectionless.
   TCP provides reliable communication, UDP doesn't and may require retransmission.

3. A uniform resource identifier (URI) is a global identifiers of network- retrievable documents (web resources). URIs adhere a uniform syntax (grammar) .

   `http://thelast.net:80/problem?course=gencs2#solution`
   scheme authority path query fragment

4. Transport layer security (TLS) is a cryptographic protocol that encrypts the segments of network connections at the transport layer, using asymmetric cryptography for key exchange, symmetric encryption for privacy, and message authentication codes for message integrity.

   (a) Client presents a list of supported encryption methods
   (b) Server picks the strongest and tells client (C/S agree on method)
   (c) Server sends back its public key certificate (name and public key)
   (d) Client confirms certificate with CA (authenticates Server if successful)
   (e) Client picks a random number, encrypts that (with servers public key) and sends it to server.
   (f) Only server can decrypt it (using its private key)
   (g) Now they both have a shared secret (the random number)
   (h) From the random number, both parties generate key material

---

4 Problem Solving and Search

Problem 4.1 (The wolf, the goat and the lettuce)

10pt

A classical children problem is the following:
A man needs to cross a river, together with his wolf, goat and lettuce. However, only himself and one of his three possessions can fit in the boat, so he needs to carry them one by one. The problem is that in the absence of the man the goat eats the lettuce and the wolf eats the goat (if left together).
Formally express this problem as a search problem (by clearly defining the 4 parameters S, O, I, G) and then give a solution.

Solution: $S = \{(left; boat; right) | left, boat, right \subset \{man, goat, lettuce, wolf\}, left \cup right \cup boat = \{man, goat, lettuce, wolf\}, left \cap right \cap boat = \varnothing\}$
$O = transferringoneelementof\{man, goat, lettuce, wolf\}atatimebetweenleft, right, boat$

$I = (man, wolf, goat, lettuce; -; -)$
$G = (-; -; man, wolf, goat, lettuce)$

Solution:
$(man, wolf, goat, lettuce;\ ;\ )$
$(wolf, lettuce;\ man, goat;\ )$
$(wolf, lettuce;\ ;\ man, goat)$
$(wolf, lettuce;\ man;\ goat)$
$(wolf, lettuce, man;\ ;\ goat)$
$(wolf;\ man, lettuce;\ goat)$
$(wolf;\ ;\ goat, man, lettuce)$
$(wolf;\ man, goat;\ lettuce)$
$(wolf, man, goat;\ ;\ lettuce)$
$(goat;\ man, wolf;\ lettuce)$
$(goat;\ ;\ man, wolf, lettuce)$
$(goat;\ man;\ wolf, lettuce)$
$(goat, man\ ;;\ wolf, lettuce)$
$(;\ goat, man;\ wolf, lettuce)$
$(;\ ;\ man, wolf, goat, lettuce)$

Problem 4.2 (A^* vs BFS)
Does A^* search always expands fewer nodes than BFS? 5pt

Solution: No. With a bad heuristic, A^* can be forced to explore the whole space, just like BFS.

---

5 Programming in Prolog

Problem 5.1 (Permutations in `ProLog`)

15pt

1. Construct a predicate eliminate(X,Y,Z) that eliminates the element X from the list Y (the result being list Z). If the element is not in the list, the predicate should yield no solution ($false$).

2. Use the predicate above to define another predicate, permute(X,Y), that computes all the permutations of list X. permute(X,Y) is true if Y is a permutation of X.

Solution:

```
eliminate(_,[],[]).
eliminate(X,[X|A],B) :- eliminate(X,A,B).
eliminate(X,[Y|A],[Y|B]) :- eliminate(X,A,B), X\==Y.

permute([],[]).
permute([H|T],Y) :- length([H|T],L), length(Y,L),
                    eliminate(H,[H|T],X1), eliminate(H,Y,Y1),
                                           permute(X1, Y1).
```
