# Seminar problems in computation

## Models of computation

1) Design either a DFA or NDFA to control a washing machine. You may include failure states if you wish.

2) In the slides is a Turing machine for recognising a language *B* = {*w*#*w* | *w* ∈ {0, 1}\* }  
Show that this machine will recognise 10#10.

## Complexity

3) Consider the following code fragment.

int *i, j*, n;

n = 100;

while (*i* < n) {

b[*i*] = *0*;

*i*++;

}

What is the time complexity and the order of time complexity of the following code? So we want an equation for the time complexity and the big O order of complexity, e.g. T(n) = 5n and O(n)

4) Consider the following recursive function.  
void f(int n) {

if (n>0) {

cout << n << " ";

f(n-1);

}

}

What is the time complexity and the order of time complexity of the following code?

5) The travelling sales person problem is a classic optimisation problem. With respect to this problem please discuss the complexity classes P, NP, NP-Complete and NP-Hard.

## Algorithms

6) *Merge sort* is a sorting algorithm that is based on the algorithmic strategy of *divide and conquer* (D&Q). Consider the following sorting algorithm, *A*, which also uses D&Q: *A* first divides a large list into two smaller sub-lists: the low elements and the high elements. *A* then recursively sort the sub-lists. The steps are:

1. Pick an element, called a *pivot*, from the list.
2. Reorder the list so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final (correct) position.
3. Recursively apply the above steps to the sub-list of elements with smaller values and separately the sub-list of elements with greater values. The base cases of the recursion are lists of sizes zero or one, which never need to be sorted.

In pseudocode:

function *A*(*array*)

if length(*array*) ≤ 1

return *array* // *array* is already sorted

select and remove an element *pivot* from *array*

create empty lists *less* and *greater*

for each *x* in *array*

if *x* ≤ *pivot* then append *x* to *less*

else append *x* to *greater*

return concatenate(*A*(*less*), [*pivot*], *A*(*greater*))

1. As examples of D&Q algorithms, what is the fundamental difference between *merge sort* and *A*?
2. As sorting methods, what are the similarities and differences between *merge sort* and *A*?

7) Consider the following alternative recursive *definition* of the factorial of a nonnegative

integer *n*:

How could this definition be turned into a recursive *algorithm*?

## Probability

8) Suppose that we have found that the word “Rolex” occurs in 250 of 2000 messages known to be spam and in 5 of 1000 messages known not to be spam. Estimate the probability that an incoming message containing the word “Rolex” is spam, assuming that it is equally likely that an incoming message is spam or not spam. If our threshold for rejecting a message as spam is 0.9, will we reject such messages?

9) In a communication channel a zero or a one is transmitted. The probability that a zero is transmitted is 0.1. Due to noise in the channel, a zero can be received as one with probability 0.01, and a one can be received as a zero with probability 0.05. If you receive a zero, what is the probability that a zero was transmitted? If you receive a one what is the probability that a one was transmitted?