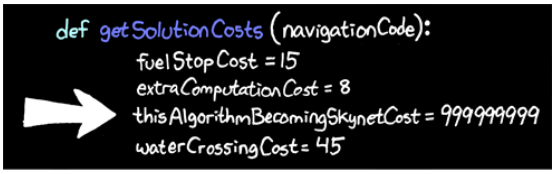
Assignment 1: GA's

Complete the programming exercises in Java, Python, R, C++ or Delphi/Pascal. Answer the questions in the report.
No introduction is needed. Do not forget to add your code to the report (as appendix to the report **and** as separate digital code files, the code files must be bundled in a zip or rar file). Send your files attached to an email to m.capalbo@maastrichtuniversity.nl, with subject: CCN Practical 1, **<your ID(s)>**

The infinite monkey theorem states that given an infinite time, a bunch of monkeys tapping keys on a typewriter will produce a meaningful text, such as a work of Shakespeare (for more information: http://en.wikipedia.org/wiki/Infinite_monkey_theorem). Here we will test the added value of evolutionary pressure to see if we can reduce the infinite time to something we can see in our lifetime (or even a few minutes).

You goal is to implement a Genetic Algorithm. The test case in our practical session is not the work of Shakespeare, but something equally important: HELLO WORLD. **Write a genetic algorithm that can evolve a population of strings to HELLO WORLD**. To get you started, I have provided a few java classes on the Student Portal. The program Practical2.java contains some code to randomly initialize a population of individuals. The class Individual is a very basic object that provides you with a structure to model your population (you are free to implement your own).



GENETIC ALGORITHMS TIP:
ALWAYS INCLUDE THIS IN YOUR FITNESS FUNCTION

I also provided a sorting class, called Heapsort.java, which (obviously) performs a Heapsort. The sorting is based on fitness of individuals. The fitness needs to be provided by the getFitness() method of the Individual and *one of your important jobs is to properly set the fitness in each Individual in each generation*. Heapsort is an inplace sorting, and extremely fast. Inplace sorting means that the array you put into it, is the same array that is returned. Except that individuals will be sorted from highest to lowest fitness value. The individual with the largest fitness value will be found at index 0, with the second highest fitness value at index 1, etc.

Your program should have at least the following components (comment in the code on what choices you have implemented):

- A selection method: you are free to choose which
- A crossover method: again, free to choose how complicated
- A mutation method: you can randomly draw a letter of the alphabet array and replace a letter in the chromosome. Keep in mind the mutation rate (the chance of a mutation per individual per generation).

When you have finished your program, play around with it by varying some parameters, and try to answer the following **questions**:

1. What is the influence of a larger/smaller population? Do you see a difference in number of generations needed to find a solution?
2. What is the influence of a larger/smaller mutation rate? Do you see a difference in number of generations needed to find a solution?
3. Now leave out the crossover operator. Do you see a difference in number of generations needed to find a solution?
4. Does your GA still work without mutation? Why or why not?
5. Questions 1 – 4 seem to suggest an optimum for this problem. What is it? Would this work for all problems? Why?

Remarks:

- The assignment is **optional not mandatory**. If you complete the assignment, it will count towards maximum 0.5 bonus points added to your exam score. 0.5 is when you score 10 out of 10 for a perfect report. All lower grades will be proportionally less bonus points.
- You are allowed to work **alone or in pairs**, not in larger groups. One pair can hand in one assignment.
- Plagiarism will be considered fraud, and fraud is sufficient reason for exclusion from the course.
- You are free to choose a different encoding of the problem if you are more comfortable with that.
- You are free to use Java, Python, R, C++ or Delphi/Pascal, no other languages
- You need to hand in code AND a report, don't forget to add your **ID number(s)**
- The code needs to run with standard dependencies, good code has comment, is readable and is 'elegant'
- **The report needs to be well written and good reports contain insights and graphs. The more it looks and feels like a 'scientific article', the better, the higher the grade**.
- **The report is due 18:00 on the Friday after the exam.**

Images from wikipedia.org and xkcd.com