**CONCORDIA UNIVERSITY**
**DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING**

**SOEN 6611: SOFTWARE MEASUREMENT: SECTION AA**
**SUMMER 2016**

# DELIVERABLE 1

## PROBLEM 1 [20%]

(a) Using the given description, construct a **use case model** for `MINSK`.

(b) Using the **use case points (UCP)** approach (or one of its extensions), provide an estimate of the effort for the project.

## NOTES

If you make any assumptions, state them explicitly.

## PROBLEM 2 [80%]

This part should use Part 1 as its basis.

- `MINSK` must be written from **scratch** using **one** of the following object-oriented programming languages: C++, Java, or Python. This means, apart from the functions related to input and output, an implementation of `MINSK` should **not** make use of any reuse mechanism (such as built-in functions, libraries, packages, or APIs) provided by the programming language.

- `MINSK` must be written from a software engineering viewpoint. In general, MINSK must aim to be **general, efficient, maintainable, portable, and usable**, as defined in Appendix A. In particular, MINSK must aim for **minimal cyclomatic number** (also known as cyclomatic complexity).

- `MINSK` must have a class called `Quadratic` to compute roots of the quadratic equation. `Quadratic` must avoid potential numerical issues related to $b^2 >> 4ac$ and $a \sim 0$ (that is, $a$ approximately equal to 0).

- `MINSK` must have a class called `Cubic` to compute roots of the cubic equation. `Cubic` must avoid potential numerical issues related to $a \sim 0$ (that is, $a$ approximately equal to 0).

- `MINSK` must have a class called `Prime` that accepts an integer as input and, as part of its output, states if it is a prime number.

**NOTES**

If you make any assumptions, state them explicitly. For example, an assumption for both `Quadratic` and `Cubic` could be that the error tolerance (that is, the magnitude of the difference between the exact and approximate values) for a root does not exceed $10^{-2}$. For example, an assumption for `Prime` could be on the size of the input.

- The use of `Quadratic` must be demonstrated for some combinations of $a$, $b$, and $c$ that are special in some way, and those that are on the 'boundary'. For example, $a = 1$, $b = 0$, $c = -2$, and $a = 1$, $b = -1$, $c = -1$, are special.

- The use of `Cubic` for some combinations of $a$, $b$, $c$, and $d$ that are special in some way. For example, $a = 1$, $b = -3$, $c = 3$, $d = -1$, and $a = 1$, $b = -6$, $c = 11$, $d = -6$, are special.

- The use of `Prime` must be demonstrated. For example, this could include at least two different inputs, one of an even integer and the other of an odd integer. (There are several interesting prime numbers[1], including Mersenne prime numbers.)

The submission must include **source code, test data, and documentation**. The documentation must include the following:

- A list of algorithm(s) used and the rationale for the choice(s) of the algorithm(s).

- An explanation of the steps taken towards achieving the quality attributes (such as citation and reference to programming style adopted, programming patterns used, as well as any other technical reasons).

- A calculation of the cyclomatic numbers of `Quadratic`, `Cubic`, and `Prime`, and an explanation as to why they are minimal.

[1] URL: http://primes.utm.edu/ .

## APPENDIX A

The quality attributes mentioned in this document vary on a spectrum.

In theory, all the quality attributes mentioned above are equally desirable. In practice, however, there are resource constraints, such as those of time and budget, which can lead to compromises between competing quality attributes.

There are no sufficient conditions for the quality attributes mentioned in this document, but there are certain necessary conditions.

### EFFICIENCY

The efficiency of a program may not only depend on the choice of the algorithm, but also on the choice of the data structure used by that algorithm.

### GENERALITY

Let $P_1$ and $P_2$ be two programs that solve the same problem. If everything possible by $P_1$ is possible by $P_2$ (but not conversely), then $P_2$ is more general than $P_1$. Usually, programs are made more general by accommodating larger class of possible inputs. For example, the generality of `Quadratic` can be seen on (at least) two dimensions: the type and the size of *a*, *b*, and *c*.

### MAINTAINABILITY

A program that takes into consideration the needs of programmers, other than the author(s) of the original program, tends to be more maintainable than the one that does not. A programmer is not necessarily the maintainer of the program.

### PORTABILITY

The submission should include only the raw source code and data files (if any). It should not depend on the use of a specific computing environment (like non-standard libraries or an IDE).

### USABILITY

A program that takes into consideration the needs of users, including their background and education level, and avoids self-referential design tends to be more usable than the one that does not. An appropriate response in case of an error made by a user can also contribute to usability. A programmer is not necessarily the user of the program.

For formal definitions and details, see the ISO/IEC 25010 Standard.