FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE CSC01A1
Introduction to algorithm development (C++)
CAMPUS APK

EXAMINATION: June/July SSA 2019

DATE: 2019-06/07

ASSESSOR(S) **PROF DA COULTER**

INTERNAL MODERATOR **MR BR GREAVES**

DURATION 3 HOURS **MARKS 100**

SURNAME, INITIALS (or ID NUMBER): _____

STUDENT NUMBER: _____

SR NR: _____

CONTACT NR: _____

NUMBER OF PAGES: 3 PAGES

REQUIREMENTS: NON-PROGRAMMABLE CALCULATORS ARE PERMITTED

Marker:			Submission overseen by:	
Sort Rank	Result	Moderation	Correction	**Submission**
				CD:
				USB:
				EVE:

Mark sheet		
Surname:		
Initials:		
Computer:		
Competency	*Description*	*Result*
C0	Program Design	/10
C1	Boiler plate code • Standard namespace (1) • System library inclusion (3) • Indication of successful termination of program (1)	/5
C2	Coding style • Naming of variables (1) • Indentation (1) • Use of comments (1) • Use of named constants / ennumerations (1) • Program compiles without issuing warnings (1)	/5
C3	Functional Abstraction • Task decomposition (5) • Reduction of repetitive code (5)	/10
C4	Separate Compilation • Header file (1) • Guard conditions (2) • Inclusion of header file (1) • Appropriate content in header file (1) • Use of programmer defined namespace (5)	/10
C5	User Interaction • Menu System (5) • Appropriate use of input, output and error streams (5)	/10
C6	Command Line Argument Handling: • Appropriately overloaded main function (1) • Handling incorrect argument counts (1) • Use of supplied arguments (3)	/5
C7	Error Handling • Use of assertions (2) • Use of conventional error handling techniques (1)	/5
C8	Pseudo-random number generation (5)	/5
C9	Dynamically allocated two dimensional arrays of structures • Structures (5) • Allocation (5) • Initialisation (5) • Deallocation (5)	/20
C10	Algorithm implementation • Logical Correctness (5) • Effectiveness / Efficiency of approach (5) • Correct output (5)	/15
B	Bonus	/10
Total:		**/100**

Markers Signature:_____

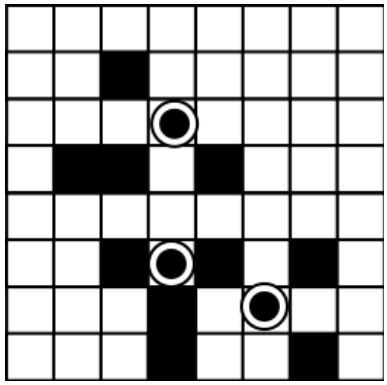
I declare that I am eligible to write this summative assessment according to the rules and regulations of the Academy of Computer Science & Software Engineering, the Faculty of Science and the University of Johannesburg. I declare that the work submitted is my own and that I have verified the correctness of my electronic submissions.

I UNDERSTAND THAT NON-COMPILING CODE CANNOT BE AWARDED A PASSING MARK

Student Signature:_____

CRYSTAL CLEAR

The Utopian Ministry for the Fourth Industrial Revolution would like you create a simulation to the crystallisation technique used within one of its automated factories for the manufacturing of consumer electronics. The crystallisation technique involves the introduction of seed material to a bath of dissolved minerals causing them to come out of solution as follows:



Seed (double circles), factory radius (large circle), crystal(black squares)

In the game you will simulate the evolution of a two-dimensional playing area. Your logic must be placed in the `CrystalSpace` **`namespace`**.

Initialisation:

- The size of the environment and number of seeds are specified via command line arguments.
- A fixed number of seeds are randomly placed in the environment. You will need to test if there is enough space in the game environment for the number of seeds given.
- There is a random amount of dissolved minerals which is an integer value which ranges between 50% to 100% of the number of squares in the game world.

Update:

- Every open space the game has the potential to become filled with crystal as follows:
  - A cumulative 10% chance for every seed within a 1 square radius, as well as...
  - ...a cumulative 5% chance for every crystal filled square in that same radius.
- Whenever a crystal square appears one unit of dissolved materials is removed from the counter.

End-game:

- The simulation ends when there are no empty spaces left or there is no more dissolved material left.

Consider the competencies as laid out in the mark sheet.

- C0 – Create a program design. Your UML must model player movement.
- C1 – Use your knowledge of basic C++ program structure and make sure to utilise the appropriate system libraries.
- C2 – Your program must be readable by human beings in addition to compiler software.
- C3 – Demonstrate your knowledge of the divide and conquer principle using functions.
- C4 – Your program must make use of programmer defined source code libraries.
- C5 – Create a menu system which will ask the user which action they wish to take.
- C6 – The user must provide the number of rows and columns used by the simulation (reasonable maxima and minima should be imposed).
- C7 – Provide assertion based error handling as well as conventional error handling.
- C8 – Random numbers are used when initialising the game world.
- C9 – Use dynamic 2D arrays and structures to implement your simulation. The data must be output to screen using printable ASCII characters.
- C10 – Pay careful attention to the updating of the world and tests for the end of the game..
- Bonus – Make use of C++11/14 features.

.