UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	CSC1A10
	Introduction to algorithm development (C++)
CAMPUS	**APK**

EXAMINATION
PAPER B

DATE: 2014-06-10

ASSESSOR(S) **MR DA COULTER & MR M HEYDENRYCH**

INTERNAL MODERATOR **MR DT VAN DER HAAR**

DURATION 3 HOURS **MARKS 100**

SURNAME, INITIALS (or ID NUMBER):______________________________

STUDENT NUMBER: ______________________________

COMPUTER 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 (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 (5)	/5
C8	Pseudo-random number generation (5)	/5
C9	Dynamically allocated two dimensional array handling • Allocation (5) • Initialisation (5) • Deallocation (5)	/15
C10	Algorithm implementation • Logical Correctness (5) • Effectiveness / Efficiency of approach (5) • Correct use of appropriate selection / iteration structures (5) • Correct output (5)	/20
B	Bonus (to be completed in a separate program)	/10
Total:		**/100**

Markers Signature:____

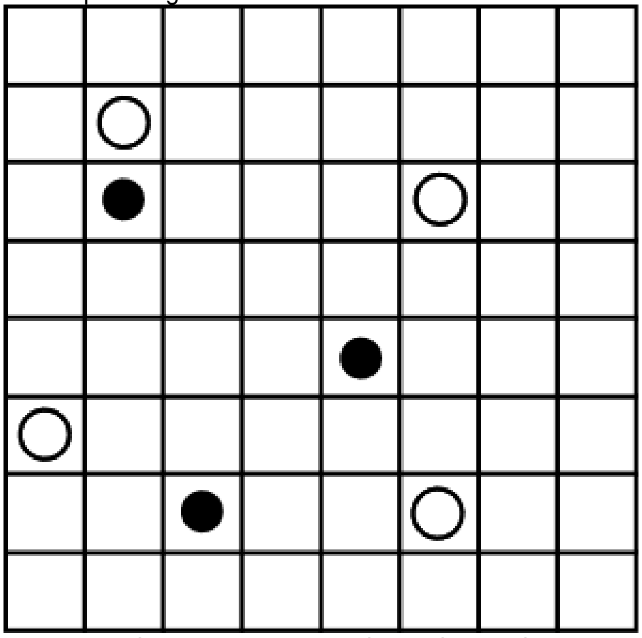
I declare that I am eligible to write this summative assessment according to the rules and regulations of the Academy of Computer Science & Software Engineering, the Faculty of Science and the University of Johannesburg. I declare that the work submitted is my own and that I have verified the correctness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

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

Student Signature:____

Smoke Signals

The Utopian Industrial Council wishes to study the effects of pollution on the atmosphere. Specifically, they are interested in examining the way that smoke produced by factories dissipates in the atmosphere. They have hired you to produce a simulation of Utopia's largest industrial sector.



Black Circle: Factory White Circle: Smoke Cloud

You will need to create a C++ simulation of smoke drifting from factories. Each patch of smoke produced has a density between 1 and 9. You must keep track of how smoke drifts. Due to the strict zoning laws in Utopia, a grid based simulation is appropriate. As well as this, wind only comes in distinct gusts in Utopia, so a turn-based simulation is appropriate. Your simulation logic must be placed in the `SmokeSpace` namespace.

Simulation initialisation:

- Exactly 11% (rounded up) of the cells in the simulation must be factories.
- There will be no smoke initially.

Simulation Progress:

- In each turn, there is a 23% chance that each factory will produce a cloud of smoke. The smoke produced will have a random density. It must be placed randomly in one of the four blocks surrounding the factory.
- Every turn, every cloud of smoke has its density reduced by one.
- If the density of a cloud of smoke reaches 0, it disappears.
- Every turn, a gust of wind blows randomly North, South, East or West. Every cloud of smoke moves in that same direction.
- Smoke may go outside of the simulation environment.
- The simulation ends after a specified number of clouds of smoke has been produced.

Using your knowledge of good software engineering principles and C++ you must design and implement such a simulation as follows. Consider the competencies as laid out in the mark sheet.

- C0 – Create a program design. Your UML must model the initialisation of the world.
- 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 from the command line (range checked based on screen width). Additionally the number of clouds to be produced before stopping must be taken from the command line.
- C7 – Provide assertion based error handling. When submitting be sure to disable assertions.
- C8 – Random numbers are used when initialising the 2D array.
- C9 – Use dynamic 2D arrays to implement your simulation (main array and scent array). The main array may be output to screen using printable ASCII characters.
- C10 – Pay careful attention to checking the legality of moves.
- Bonus – Produce an SDL based visualisation of the simulation.