UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

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

EXAMINATION
SSA

DATE: 2014-07

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.

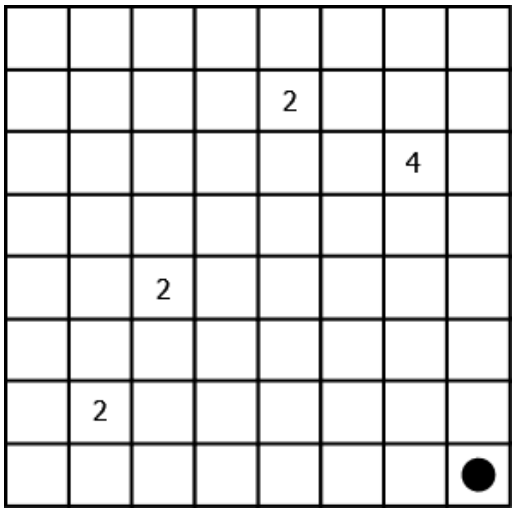
I UNDERSTAND THAT NON-COMPILING CODE CANNOT BE AWARDED A PASSING MARK

Student Signature:_____

.

Warehouse 2048

The Utopian Warehouse Complex has recently designed a robot for stacking crates in their warehouse. They have commissioned you to produce a program which will be used for controlling the robot. The robot pushes crates, and when two crates of the same size are pushed together, the robot automatically stacks the crates together.



Robot (black circle), Crate (number)

You will need to create a C++ simulation of the warehouse. The manager is interested in finding the most efficient stacking of crates. Due to the way in which commands are sent to the robot, a turn based simulation is appropriate. Your simulation logic must be placed in the `StackSpace` namespace.

Simulation initialisation:

- The robot starts in the bottom left corner of the map
- Crates cannot be initialised to the edges of the warehouse
- Each legal block has a 13% chance of having a box of size 2

Moving:

- The robot may move up, down, left or right
- If the robot attempts to move into a block currently occupied by a crate, there are three possibilities
  - If there is an open block next to the crate in the same direction as the robot moved, the crate will be moved into that block.
  - If there is a crate with the same value in the same direction as the robot moved, the two crates will be stacked. In this case, they will become a single crate with a size which is the sum of the crates.
  - If there is a crate with a different value in the same direction as the robot moved, nothing will happen.
- Crates may not be pushed out of the warehouse, and the robot may not move out of the warehouse.
- The simulation ends when there is only one crate left. Since this may not be possible, quitting must be an option.

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 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 (range checked based on screen width).
- 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. The 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.

.