University of Westminster
School of Computer Science and Engineering

5SENG003W	Algorithms – Referred/deferred Coursework (2022/23)
Module leader	Klaus Draeger
Unit	Coursework
Weighting:	50%
Qualifying mark	30%
Description	Solving sliding puzzles using path finding
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): - LO2: Be able to apply the theory for the effective design and implementation of appropriate data structures and algorithms in order to resolve the problem at hand; - LO3: Be able to analyse, predict, compare and contrast the performance of designed and implemented algorithms, particularly in the context of processing data; - LO4: Be able to use a range of typical data structures and collections as part of Application Programming Interfaces (APIs) offered by programming languages; - LO5: Be able to apply the theory for the definition and implementation of novel algorithms.
Handed Out:	26th May 2023
Due Date	13:00, 14th July 2022
Expected deliverables	**A zip file containing the source code in Java or C++,** **a short report (no more than 3 pages pdf).**
Method of Submission:	Electronic submission on Blackboard via a provided link close to the submission time.
Type of Feedback and Due Date:	Written feedback within 15 working days.
BCS CRITERIA MEETING IN THIS ASSIGNMENT	**2.1.1 Knowledge and understanding of facts, concepts, principles & theories** **2.1.3 Problem solving strategies** **2.1.5 Deploy theory in design, implementation and evaluation of systems** **2.2.2 Evaluate systems in terms of quality and trade-offs** **2.3.2 Development of general transferable skills** **3.2.2 Defining problems, managing design process and evaluating outcomes** **4.1.1 Knowledge and understanding of scientific principles** **4.1.2 Knowledge and understanding of mathematical and statistical principles** **4.2.1 Use theoretical and practical methods in analysis and problem solving**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

Coursework Description: Sliding puzzles

In this coursework, you are supposed to use path finding to solve a type of puzzle that occurs in many video games. An example for the basic version we will be dealing with is this:

```
S. . .
0. F.
. . . .
. . . 0
```

The player starts at the location labelled "S" and wants to reach the finish, labelled "F". Each turn they choose one of the four cardinal directions (up/down/left/right) to move. However, except for S and F the floor is covered in ice, so they will keep sliding in the chosen direction until they hit the wall surrounding the area, or one of the rocks (labelled "0"). For example, starting in the map given above:

```
@. . .
0. F.
. . . .
. . . 0
```

the player ("@") moving right would end up here:

```
S. . @
0. F.
. . . .
. . . 0
```

We are dealing with the problem of finding a path from S to F, but the reachability relation between points is not the usual one.

Tasks to be performed:

Task 1 (10 marks). Set up a project (Java or C++) as you did for the tutorial exercises.
Task 2 (20 marks). Choose and implement a data structure which can represent maps such as the one in the example. It must provide the necessary infrastructure for finding a shortest path from the start to the finish.
Task 3 (20 marks). Add a simple parser which can read a map like the one in the example from an input file. It needs to determine the width and height and the locations of the start and finish square as well as the rocks. The structure of the files will look like in the example, i.e., use '.'/'0'/'S'/'F' for empty (ice) squares, rocks, the start, and the finish.
Your parser should be able to handle all input files which have this format. We will provide benchmark examples for your performance analysis, but you may also want to create some yourself to test your implementation.
Task 4 (20 marks). Choose and implement an algorithm which finds a shortest path from the start to the finish in any given map, if one exists (all the benchmarks we provide will have a solution). It should output all the steps of the solution it found, e.g., for the example above:

1. Start at (1,1)
2. Move right to (4,1)
3. Move down to (4,3)

4. Move left to (1,3)
5. Move down to (1,4)
6. Move right to (3,4)
7. Move up to (3,2)
8. Done!

Where the squares are numbered left to right, top to bottom.

Task 5 (30 marks). Write a brief report (no more than 3 A4 pages) containing the following:

a) A short explanation of your choice of data structure and algorithm.
b) A run of your algorithm on a small benchmark example. This should include the supporting information as described in Task 4.
c) A performance analysis of your algorithmic design and implementation. This can be based either on an empirical study, e.g., doubling hypothesis, or on purely theoretical considerations, as discussed in the lectures and tutorials. It should include a suggested order-of-growth classification (Big-O notation).

To be submitted:

- Your zipped source code (for Tasks 1 to 4) in Java or C++. Your source code shall include header comments with your student ID and name.
- The report about the algorithmic performance analysis (Task 5).

Coursework marking scheme:

Criterion and range	Indicative mark	Comments
Task 1 (0-10 marks)	8-10	A compilable and executable project has been created and follows guidelines for writing clear code such as https://introcs.cs.princeton.edu/java/11style
	4-7	A compilable and executable project has been created but does not follow programming guidelines such as of the *Java* related example above.
	0-3	No project has been created, or it is prone to compilation or runtime errors.
Task 2 (0-20 marks)	16-20	A data structure has been implemented, which satisfies the following principles of abstraction: - Builds on top of already existing programming language specific data structures, e.g., array. - Allows maps as given by the input to be represented - Fits the purpose of the intended algorithm and nature of the problem.
	5-15	A working data structure has been implemented but is limited in functionality
	0-4	A data structure has not been implemented or does not work properly.
Task 3 (0-20 marks)	16-20	A parser has been implemented and is able to initialise the data structure from a given input file. It can handle any input file which has the format given in the problem description.
	5-15	A parser has been implemented and is able to initialise the data structure from some input files, but not others.
	0-4	A parser has not been implemented or does not work properly.
Task 4 (0-20 marks)	16-20	The correct solution, i.e., a shortest path from start to finish, can be found for any solvable problem instance. The implementation outputs the steps of the solution in enough detail that it can be checked independently.
	5-15	The correct solution can be calculated, but the implementation does not work for all possible inputs, or does not provide enough information to justify its result.
	0-4	Not done, or does not work properly
Task 5 (0-30 marks)	25-30	The student has submitted a full report explaining their solution and its algorithmic performance based on sufficient empirical data or a formal analysis
	6-24	The student has submitted a report, but some of the contents (explanation of the data structure, explanation of the algorithm, discussion of algorithmic performance) are lacking.
	0-5	Not done, or no relevant contents.