University of Central Lancashire
UCLan

Course: BSc Computing	**Module Code:** CO1406 **Module Title**: Data Structures and Algorithms
Title of the Brief: Theseus vs. Minotaur	**Type of assessment**: [report, portfolio]

This assessment is worth 60% of the overall module mark.

This Assessment Pack consists of a detailed assignment brief, guidance on what you need to prepare, and information on how class sessions support your ability to complete successfully. You'll also find information on this page to guide you on how, where, and when to submit. If you need additional support, please make a note of the services detailed in this document.

How, when, and where to submit:

The assessment release date is: 07/11/2023.

The assessment deadline date and time is: **19/03/2024 @23:59**

Feedback will be provided by: 12/04/2024.

You should aim to submit your assessment in advance of the deadline.

Note: If you have any valid mitigating circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you will need to apply online, via MyUCLan with your evidence **prior to the deadline**. Further information on Mitigating Circumstances via this link.

SUBMISSION DETAILS

You are required to submit the following:

1. Add the completed assignment coversheet (see page 4)
2. All deliverables mentioned in section [Deliverables] of the **Detailed Assignment Brief**

We wish you all success in completing your assessment. Read this guidance carefully, and if any questions, please discuss with your Module Leader.

Additional Support available:

All links are available through the online Student Hub

1. Academic support for this assessment will be provided by contacting ***Panayiotis Andreou*** pgandreou@uclan.ac.uk.

2. Our **Library resources** link can be found in the library area of the Student Hub or via your subject librarian at CyprusLibrary@uclan.ac.uk

3. For help with Turnitin, see Blackboard and Turnitin Support on the Student Hub

4. If you have a disability, specific learning difficulty, long-term health or mental health condition, and not yet advised us, or would like to review your support, **Student Support Officers (cyprusstudentsupport@uclancyprus.ac.cy)** can assist with reasonable adjustments and support. To find out more, you can visit the **Student Support Service** page of the Student Hub. You can also call +357 24694026 or +357 24694108 or +357 24694073.

5. For mental health and wellbeing support, please complete our online referral form or email the Psychological Wellbeing and Counselling Centre at UCLan Cyprus at wellbeing@uclancyprus.ac.cy.

6. For any other support queries, please contact **Student Support** via CyprusStudentSupport@uclan.ac.uk.

7. For consideration of Academic Integrity, please refer to detailed guidelines in our policy document . All assessed work should be genuinely your own work, and all resources fully cited.

8. For advice on the use of Artificial Intelligence, please refer to Categories of AI tools guidance.

For this assignment, you are not permitted to use any category of AI tools.

Preparing for your assignment.

Refer to the Module Information Pack to understand the Learning Outcomes and Marking Criteria.

University of Central Lancashire
UCLan

Feedback Guidance:

Reflecting on Feedback: how to improve.

From the feedback you receive, you should understand:

- The grade you achieved
- The best features of your work
- Areas you may not have fully understood
- Areas you are doing well but could develop your understanding.
- What you can do to improve in the future - feedforward

Next Steps:

- List the steps have you taken to respond to previous feedback.
- Summarise your achievements
- Evaluate where you need to improve here (keep handy for future work):

University of Central Lancashire
UCLan

Coursework Cover Sheet

Students should complete the input fields contained in this form and attach it in front of their formal assessment submission. All fields within this form are required. Please ensure that checkboxes and radio buttons are appropriately selected. The last three questions are just for you to personally consider.

Department and assessment information:

School Name: Enter the school name here.
Assessment title: Enter the title of the assessment here
Course Title: Enter the Course Title here
Module Title: Enter the Module Title here
Module Code: Enter the Module code here
Year of Study: Enter your year of study eg. 2023

Academic Misconduct / Plagiarism Declaration

By attaching this front cover sheet to my assessment I confirm and declare that **I am the sole author of this work**, except where otherwise acknowledged by appropriate referencing and citation, and that I have taken all reasonable skill and care to ensure that no other person has been able, or allowed, to copy this work in either paper or electronic form, and that prior to submission I have read, understood and followed the University regulations as outlined in the Academic Integrity Policy and Procedure for Academic Misconduct

Have you checked the following? This will help your assessment achievement.

I have applied the learning outcomes for this module ☐
I have checked for Academic Integrity via Turn-it-in ☐
I have followed the guidance in the Assessment Brief and have not used AI to boost my grade unfairly. ☐
I have used references in accordance with instructions in the Assessment Brief ☐
I have proofread my work for spelling, grammar and punctuation. ☐
I have checked that the word count/size of this submission accords with the guidance provided in the Assessment Brief.☐

Well-being

We wish to support any student who is experiencing mitigating circumstances which prevent students from performing to the best of their ability when completing or submitting assignments. If you are experiencing such circumstances, then you may apply for Mitigating Circumstances. Wherever possible this must be done prior to handing in your assignment.

Do you need to apply for mitigating circumstances for this assignment Please select Yes / No

Please refer to the Mitigating Circumstances Policy

Questions you may wish to consider:

1. Have I allowed sufficient time to prepare this assessment?

2. Have I reflected on previous feedback and made improvements in accordance with advice?

3. What grade am I expecting?

University of Central Lancashire
UCLan

Detailed Assignment Brief

IMPORTANT

- **Read the marking scheme carefully**.
- **This is an individual project** and no group work is permitted.

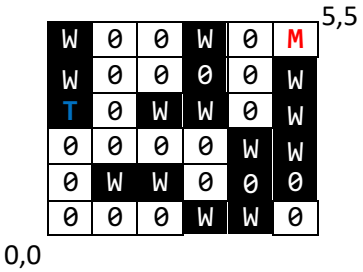
Learning Outcomes

- To develop problem-solving skills
- To evaluate common data structures
- To enhance practical programming skills.

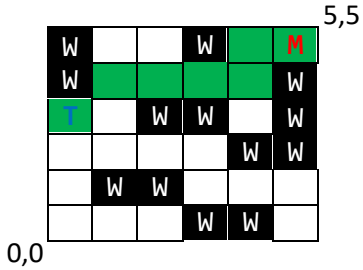
Assignment Purpose and Overview

The purpose of this assignment is to implement a non-recursive backtracking algorithm to help Theseus (https://en.wikipedia.org/wiki/Theseus) defeat the Minotaur. Minotaur was a fearsome beast with the head of a bull and the body of a man, which lived in a vast underground maze built by Daedalus under instructions from Minos, the king of Crete.
Your objective is to implement an algorithm that will assist Theseus to navigate through the maze, find the Minotaur and attempt to defeat him. If Theseus succeeds in defeating the Minotaur, then he will need to escape the maze.

The maze can be represented by a rectangle of an arbitrary number of rows and columns. In the following figure, we provide a maze example with 6 rows and 6 columns, with positions starting from (0,0), indicating the bottom left cell, and (5,5), indicating the top right cell. "T" indicates the entrance to the maze, where Theseus is located at coordinates (3,0), and "M" indicates the location of the Minotaur in the maze at coordinates (5,5). Maze walls are provided with the letter "W" and empty corridors with the digit "0".



In the first step, the program needs to discover the route (if it exists) to the Minotaur. Using the above maze example, a possible route for Theseus to find the Minotaur can be shown in the following figure (using green colour for the route).



The output of the first step should be the coordinates (row, column) of the steps required to reach the Minotaur's location, which can be seen below.

(3,0) T
(3,1)
(4,1)
(4,2)
(4,3)
(4,4)
(5,4)
(5,5) M

In the second step, Theseus will attempt to defeat the Minotaur using his sword. In this fictional scenario, Theseus will have a 70% probability to defeat the Minotaur and 30% to fail. The output of this step should be:

Theseus defeats the Minotaur, or
Minotaur defeats Theseus.

In the case that Minotaur defeats Theseus the program will need to stop. Otherwise, you will need to proceed to the third step.

In the third step, the program will need to provide a route for Theseus to escape the maze. The output of this step can be the following:

(5,5)
(5,4)
(4,4)
(4,3)
(4,2)
(4,1)
(3,1)
(3,0)

For the purposes of this assignment, all maze configurations will be stored as text files in the format presented below, where:

- the first line contains the number of X rows in the maze
- the second line contains the number of Y columns in the maze
- the third line contains the position of Theseus
- the fourth line contains the position of the Minotaur
- the remaining numbers, presented as X rows of Y columns each, represent the maze configuration, as described above

Using the aforementioned example, the maze configuration will be stored as follows:

```
6
6
3 0
5 5
W 0 0 W 0 M
W 0 0 0 0 W
T 0 W W 0 W
0 0 0 0 W W
0 W W 0 0 0
0 0 0 W W 0
```

Overall, in this assignment, you will:

- Evaluate and implement **efficient data structures** for storing maze configurations
- Design an **efficient algorithm** that assists Theseus to find and defeat the Minotaur and escape the maze using **non-recursive backtracking**
- Design and implement **appropriate data structures for supporting backtracking**
- Develop a **program that orchestrates the algorithm and data structures** to solve the Theseus maze problem using backtracking

Assignment Requirements and Constraints

The assignment must start by defining **efficient data structures** to load a maze configuration. To that end, you will define the **`Maze`** data structure and will use a function to initialize an instance of the data structure accordingly. In particular, the `Maze` data structure must be able to preserve the following important information:

- number of rows and columns in the maze
- the position of Theseus
- the position of the Minotaur
- the maze configuration in a two-dimensional array

As soon as the file is loaded, the Maze data structure needs to support printing of the maze configuration in the following format:

```
+-----------------------------------------+
|W                W               M|
|W                                W|
|T          W     W               W|
|                       W         W|
|     W     W                       |
|                 W     W           |
+-----------------------------------------+
Theseus: (3,0)
Minotaur: (5,5)
Number of walls: 13
Number of corridors: 21
```

The signature of the loading function must be the following:

```
Maze* loadMaze(string filename)
```

A number of maze configurations are provided in **Appendix 1** for you to test your implementation.

The program should then proceed to find a route to the Minotaur. The algorithm will employ a backtracking approach according to the following high-level concept:

- Test if Theseus can visit a cell, i.e., the cell is a corridor and not a wall
- Visit that cell and continue to the next one using some strategy
- If you have reached a dead end, you will need to backtrack to an appropriate cell to try a different route
- Continue to visit cells until you reach the Minotaur and then attempt to defeat him

- If you defeat the Minotaur, print the appropriate message and then proceed to find an escape route
- Otherwise, stop the program and print the appropriate message

The following partial function signatures must be used; however, they can be expanded to support other operations.

- `Coordinates* findMinotaur(Maze* maze)`
- `boolean defeatMinotaur()`
- `void escapeMaze(Maze* maze)`

It is compulsory that you implement a Stack data structure to enable the backtracking functionality. The Stack data structure must be designed using generic programming techniques and should implement the following functions:

- isEmpty: checks if the stack is empty
- push: inserts an element to the top of the stack
- pop: removes an element from the top of the stack, if it exists, without returning it
- top: returns the element located on the top of the stack, if it exists

IMPORTANT: The program must satisfy all the requirements and constraints described in this section. However, the requirements may not be sufficiently defined. During the specifications, you will need to record your assumptions and how these have influenced your data structure design and algorithm implementation.

Deliverables

For part A, the deliverable consists of a compressed folder, named as "202324.CO1406.<GNumber>.zip" (e.g., 202324.CO1406.G1234567.zip), with the following files:

1. 1x Document (Microsoft Word format) consisting of the documentation of parts A-D
2. 1x compressed file (.zip) including your source code (C++ Header files (.h) developed for part B and C++ Source files (.cpp) developed for parts C and D) and any resources (e.g., the maze files) used in your implementation. You should not submit files related to your IDE (e.g., solution or project files used by Visual Studio).
3. 1x a C++ source code file (.cpp) file, including all your code, named 202324.CO1406.<GNumber>.cpp.

Failure to comply with the above requirements will result in getting 0 marks for your assignment.

Source files that do not compile will result in getting 0 marks for your assignment.

Grading Criteria

Marks will be awarded based on the following criteria. Within each part, aim to complete the work for each section before moving on to the next as you will not get the full credit for later sections if there are significant defects in an earlier section. **However, do not simply stop if you are stuck on one part, ask for feedback.**

In assessing the work within a section, factors such as simplicity, quality and appropriateness of comments, and quality and completeness of the design will be considered.

Part	Description	Range	Deliverable
A	Documentation	0-10	• Documentation in Microsoft Word (.docx) format
B	Data Structures	0-20	• C++ code for supporting data structures required for backtracking • C++ code for loading and manipulating maze configurations
C	Algorithm	0-20	• C++ code for finding routes using backtracking
D	Program	0-10	• C++ code to orchestrate deliverables of parts B to solve the problem and provide basic error control

Detailed Marking Scheme

Part	Description	Criteria
A	Documentation	0-4 **Requirements Analysis** • 0 - no attempt or Fail • 1 - Requirements are poorly described • 2 - Requirements are sufficiently described. Develops requirements for a decent solution • 3 - Provides evidence of investigative skills in problem analysis to develop requirements for an efficient solution • 4 - Identifies several potential solutions and justifies the selection of the most efficient solution. 0-4 **Complexity Analysis** • 0 - no attempt or Fail • 1-4 Evaluate the time complexity of the loadMaze, findMinotaur, defeatMinotaur and escapeMaze functions correctly 0-2 **Source code Documentation** • 1 - Function comments, which describe the functionality of a method/field before the function definition • 1 - Inline comments, which describe implementation decisions within a method body
B	Data Structures	0-2 Implementation of the **Coordinates** data structure representing a position in the maze. 0-8 Implementation of the **Maze** data structure. Implements appropriate members for storing: • 2 - number of rows and columns in the maze • 2 - the maze configuration in a two-dimensional array • 2 - the positions of Theseus and the Minotaur • 1 - the number of walls in the maze • 1 - the number of corridors in the maze

		0-4 Implementation of a **Stack** data structure for storing moves (i.e., positions) during backtracking • 1 - Development of an appropriate push method • 1 - Development of an appropriate pop method • 1 - Development of an appropriate top method • 1 - Application of Generic programming 0-4 Implementation of the **MazeSolver** data structure representing a structure with functionality to support the operations of finding the minotaur, defeating him and finding an escape plan. Implements appropriate members for: • 1 – storing the maze configuration as this was defined above • 1 - a stack to be used for backtracking • 1 - the functions required to support the operations • 1 - the number of steps required by the solver to find the Minotaur 0-2 Demonstration of selecting **Storage Efficient** data members for the data structures described above. • 0 - no attempt • 1 - Consideration of trivial or infeasible alternatives • 2 - Careful consideration of alternatives and justification of the selection of data member types
C	Algorithm	0-20 **Algorithm** • 0-10 – utilizes backtracking to find a correct route to the Minotaur; provides appropriate message when there is no route • 0-3 – prints the route when there is one including the number of steps • 0-3 – attempts to defeat the minotaur (you should use a random number generator for this step and ensure that there is a probability of 70% for Theseus to defeat the Minotaur) and prints the appropriate message • 0-4 – prints the escape plan
D	Program	0-8 **Maze Loading Function** • 0-6 initializes a Maze data structure appropriately • 0-2 checks for inconsistencies/errors in the file 0-2 **Main function** • 1 - Uses the loadMaze function to load a maze from a file to a Maze data structure • 1 - Initializes a MazeSolver data structure, loads the Maze and then invokes the solver algorithm

University of Central Lancashire
UCLan

Submission of assignment work

- Anonymous marking is being used. Apart from your University ID number ("G2…"), avoid doing anything that would allow you to be identified from your work.
- *Keep a complete copy of the work you hand in.*
- Avoid submitting work at the last minute, but if there is a technical problem uploading to Blackboard, email the zip file to me before the deadline and upload the work when Blackboard is available.

Conformance to Academic Regulations

Students and their assessments are subject to the academic regulation, which specifies what is considered an Unfair Means to Enhance Performance: Cheating, Plagiarism and Collusion. Students are expected to conform to these regulations, or face disciplinary actions, as described in Appendix 9 of the Academic Regulations (the relevant excerpts clarifying the unfair means are listed below):

Cheating

The term cheating includes, inter alia:

a. Being in possession of notes, 'crib notes', or texts books during an examination other than an examination where the rubric permits such usage;
b. Copying from another candidate's script or work;
c. Communicating during the examination with another candidate;
d. Having prior access to the examination questions unless permitted to do so by the rubric of the examination;
e. Substitution of examination materials;
f. Unfair use of a pocket calculator;
g. Impersonation;
h. Use of a communication device during the examination;
i. Or any deliberate attempt to deceive.

Plagiarism

Material submitted for assessment through open book examination, coursework, project or dissertation must be the student's own efforts and must be his/her own work. Students are required to sign a declaration indicating that individual work submitted for assessment is their own.

Copying from the works of another person constitutes plagiarism, which is an examination offence. Brief quotations from the published or unpublished works of another person, suitably attributed, are acceptable. Every School issues guidelines on the use and referencing of quotations which students are required to follow.

Collusion

Collusion is an example of unfair means because, like plagiarism, it is an attempt to deceive the examiners by disguising the true authorship of an assignment, or part of an assignment. Its most common version is that student A copies, or imitates in close detail, student B's work with student B's consent. But it also includes cases in which two

or more students divide the elements of an assignment among themselves, and copy, or imitate in close detail, one another's answers.

It is an unfair means of offence to copy, or imitate in close detail, another student's work,even with his or her consent (in which case it becomes an offence of collusion). It is alsoan offence of collusion to consent to having one's work copied or imitated in close detail. Students are expected to take reasonable steps to safeguard their work from improper use by others. Where a student is found to have engaged in collusion, the same penalties as for plagiarism will apply.

Where it is established that student B has not engaged in plagiarism, the requirement for resubmission may be waived in the case of student B.

Collusion should not be confused with the normal situation in which students learn from one another, sharing ideas, as they generate the knowledge and understanding necessary for each of them to successfully and independently undertake an assignment. Nor should it be confused with group work on an assignment where this is specifically authorised in the assignment brief.

Commissioning of Assessed Work

Commissioning occurs when a student commissions a third party to complete all or part of an assessed piece of work and then submits it as their own. Commissioned workmay be pre-written or specifically prepared for the student. It might be obtained from a company or an individual and may or may not involve a financial transaction. It includes the use of essay mills or buying work online or the use of a proof-reading service that includes re-writing the original assessed piece of work. Where it is suspected that a student has submitted work that has not been written by them, the student may be asked questions about the work during an interview with the AcademicIntegrity Officer or Academic Misconduct Committee to give them the opportunity to demonstrate appropriate knowledge of the subject matter and that they understand the content of the work. Students must keep copies of drafts and other materials usedin researching and preparing the work.

Category 4: Gross Academic Misconduct

Category 4 will normally be defined as gross academic misconduct where a clear intent to deceive and gain an unfair academic advantage can be established. Examples of category 4 gross academic misconduct include, without limitation:

- A repeat instance of category 3 academic misconduct in any form
- Commissioning of assessed work
- Fabrication or falsification of data

Academic Penalties for Category 4 offense

Level failed and a requirement to withdraw from the programme. (This does not preclude the student from applying for re-admission to the University after a period of time defined by the Committee.)
Or

Expulsion from the University on a permanent basis. The Academic Misconduct Committee will advise the Assessment Board regarding the student's entitlement to any exit award or credit achieved. The student will normally be entitled to retain an exit award or any credits awarded for work that has already been passed without evidence of academic misconduct. A flag will be placed on the student record system.

The complete set of academic regulations can be accessed online at:
http://www.uclan.ac.uk/study_here/student-contract-taught-programmes.php

Reassessment and Revision

Reassessment in written examinations and coursework is at the discretion of the Course Assessment Board and is dealt with in accordance with university policy and procedures.

University of Central Lancashire
UCLan

Appendix 1

Example file 1

```
6
6
3       0
5       5
W       0       0       W       0       M
W       0       0       0       0       W
T       0       W       W       0       W
0       0       0       0       W       W
0       W       W       0       0       0
0       0       0       W       W       0
```

Example file 2 (no solution)

```
6
6
3       0
5       5
W       0       0       W       W       M
W       0       0       0       0       W
T       0       W       W       0       W
0       0       0       0       W       W
0       W       W       0       0       0
0       0       0       W       W       0
```

Example file 3

```
6
7
0       0
0       6
W       0       0       W       0       0       W
W       0       0       0       0       W       W
0       0       W       W       0       0       W
0       0       0       0       W       0       W
0       W       W       0       0       0       W
T       0       0       W       W       0       M
```

Example file 4

```
6
7
5       1
0       6
W       T       0       W       0       0       W
W       0       0       0       0       W       W
0       0       W       W       0       0       W
0       0       0       0       W       0       W
0       W       W       0       0       0       W
0       0       0       W       W       0       M
```