College of VE – Future Technologies

COSC 2511: Introduction to Programming A

Programming Project

Due End of Week 15

Assessment weighting: This assessment is out of 100 and will contribute towards 30% of your overall course mark. (This is not a hurdle assessment)

Submission:

- This is a group project. Students must work in a group of 3. You must self-enrol into a group by the start of Week 11. After this time, groups will be locked.
    - You may choose who you would like to work with. If you do not have a group, then one will be assigned to you.
- Only one submission will be made per team.
- The due date for this assessment is the end of week 15.
- Submissions will be made via Canvas.
- AI TOOLS CANNOT BE USED IN THE COMPLETION OF THIS ASSESSMENT TASK.
- In this assessment task, you must not use any AI tools (excluding text editing software e.g., Grammarly) to generate any materials, content or ideas related to the task.
- After you have submitted your project, your team will be required to attend a mandatory meeting with your teacher to demonstrate your project.
    - During this meeting you will be graded on your submission.
    - Your teacher will ask you questions to validate your understanding of the content written for the project. It is each member's responsibility to understand the project completely.
    - Marks can be deducted at the discretion of the teacher if you fail to answer questions asked.
    - IMPORTANT: If you do not attend this meeting, you will receive a mark of 0 for this assessment.

Assessment policy:

- Assessments submitted between 0-24 hours after the due date: 20% penalty.
- Assessments submitted between 24-48 hours after the due date: 50% penalty.
- Assessments submitted more than 48 hours after the due date will not be accepted.

What to Submit:

Your team should make one single submission in a .zip file named:

- GroupName+Number.zip
    - This will be based on the group name + number you have assigned yourselves on canvas.

Your zip file should include the following:

- Your team's design document containing all the requirements outlined in the section: **What to complete**
- Your team's completed java project including all required program files submitted as .java source code files. Keep the package/folder structure in your submission.

Extensions:

- Extensions will only be granted under exceptional circumstances and are intended to offer support and flexibility where unforeseen events have occurred preventing students from submitting projects on time.
- If an extension is required, project teams must apply via email to their lab teacher prior to the due date with an explanation of the unforeseen circumstances experienced.
- If an extension is granted by your teacher, it will be for a maximum of 7 calendar days.
- If further extensions are required, all project team members must individually apply for RMIT Special Consideration

Overview

This project is designed to give you an opportunity to exercise your algorithmic thinking and Java programming skills to solve programming problems with tools practiced throughout the semester in Introduction to Programming COSC2511.

You will demonstrate every topic learned throughout the semester in this assessment.

Team Composition:

You will form teams of 3 students in Canvas via the “People” link under the “Project Groups Tab”. Select a group from your section and self -enrol along with the people you would like to work with.

If you are not sure how to access this, please ask your teacher in class.

You must allocate yourself to a team by the start of week 11. Any students not forming a team by this date will be allocated to a team.

What to Create (Core Requirements):

The topic of this project is that your submission should be a console-based text adventure game written in Java and should show an understanding of topics covered in Introduction to Programming COSC2511.

Full creative control is with your project team, and the story line is completely up to you! You might be exploring an alien planet, attacking a castle, or trying to make your way out of a spooky forest. Let your imagination run wild!

Your game environment should be based on a 5 x 5 grid layout (see visual example below) and should have a minimum of 9 locations that a player can visit throughout the game.

Your game must have a clear objective (to win the game) and potentially may have an alternate ending (optional) such as the player dying. Your game must include an inventory system allowing a player to pick up, carry, list and interact with a minimum of 5 items, and must include at least one interaction with a non-player character (npc).

Players must be able to navigate the game using a simple control system based on the four main points of a compass (north(n), south(s), east(e) & west(w)), and each time a location and any available items or interactions is visited a description of the location must be printed to the screen.

Once an item has been picked up, or a foe vanquished, subsequent visits to that location must omit the item or npc from the description (or in the case of a slain enemy, change its state to dead in the description).

Your goal is to display an understanding of the following programming tools used throughout your game.

What to complete

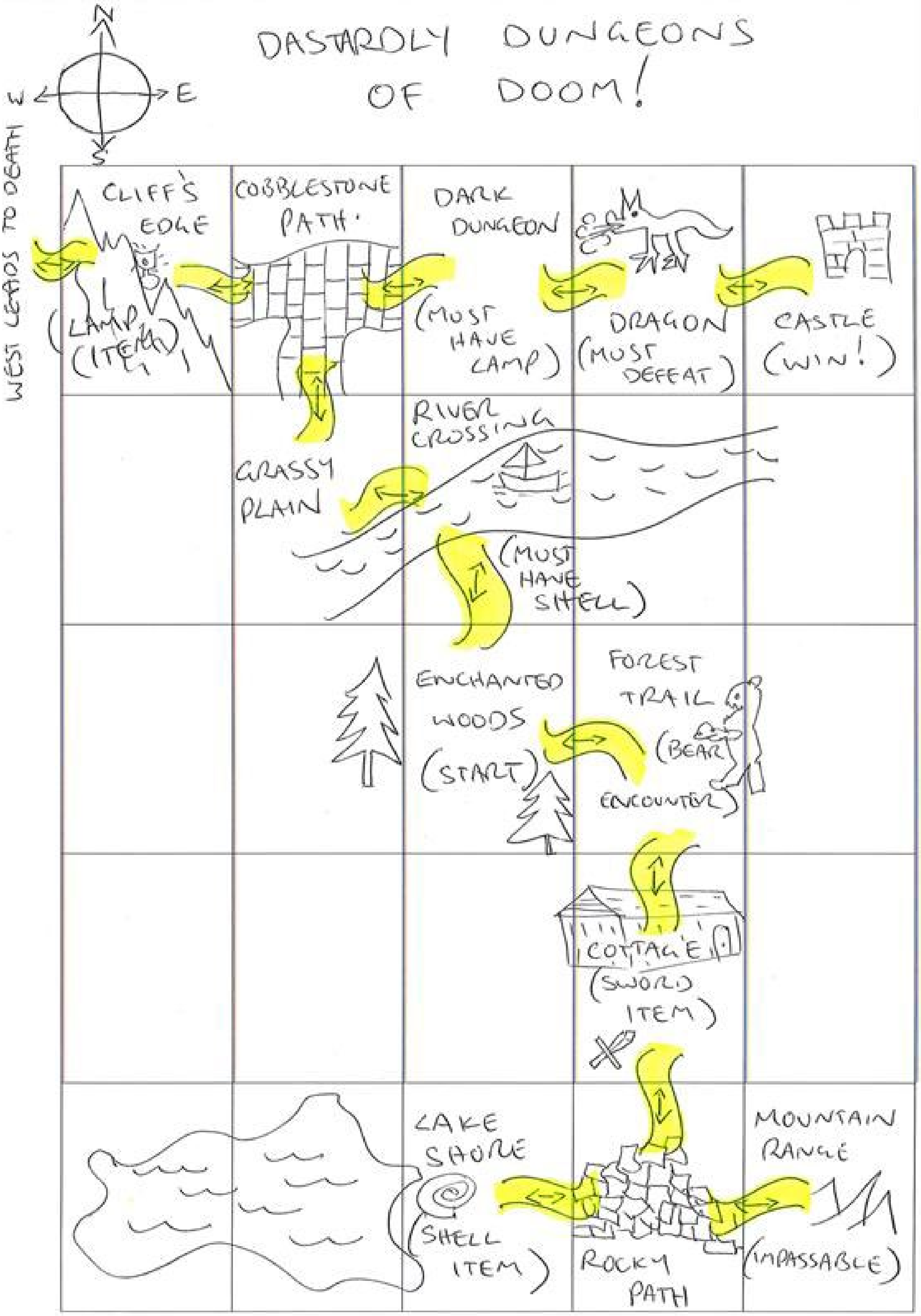
A comprehensive design document which includes:

- Overview of the game including:
    - Objective
    - Gameplay Instructions
    - Map (see next page)
- Overview of each class created **OR** completed Javadoc outlining what each class does
- Algorithms for each method created **OR** completed Javadoc outlining what each method does
- A continuous bug report – including how each issue was overcome
- Allocation of duties for each team member

A completed application which contains the following:

- Exemplary coding etiquette showing good indenting, consistent block bracing style, class naming conventions
- Appropriate use of comments.
- User (player) input
- Appropriate selection of and use of variables
- Selection statements including switch, if, else if and nested if & Implementation of user menu
- Iteration with the appropriate type of loops
- Use of random numbers
- Arrays
- Methods
- Classes

Game Map Example



Marking Guide – Documentation 35 % of the total mark

Game Objective	3 Marks	2 Marks	1 Mark	0 Marks	
	Clearly defined game objective that is challenging but achievable	Demonstrated defined game objective present but either too challenging or too easy to achieve	Demonstrated game objective present but not clear or achievable	Demonstrated no clear objective defined to win the game	
Gameplay Instructions	3 Marks	1.5 Mark	0 Marks		
	Clearly defined instructions that are concise but easy to follow	Instructions provided provide too much/little information	Instructions are not provided		
Map	5 Marks	3 Marks	2 Marks	1 Mark	0 Marks
	All the following criteria are met: -Map created is a 5X5 grid. -All locations are clearly identified. -Encounters and items are clearly identified. -All requirements are clearly labelled (such as must have lamp)	One of the following criteria is not met: -Map created is a 5X5 grid. -All locations are clearly identified. -Encounters and items are clearly identified. -All requirements are clearly labelled (such as must have lamp)	Two of the following criteria are not met: -Map created is a 5X5 grid. -All locations are clearly identified. -Encounters and items are clearly identified. -All requirements are clearly labelled (such as must have lamp)	Three of the following criteria are not met: -Map created is a 5X5 grid. -All locations are clearly identified. -Encounters and items are clearly identified. -All requirements are clearly labelled (such as must have lamp)	More than three of the following criteria are not met: -Map created is a 5X5 grid. -All locations are clearly identified. -Encounters and items are clearly identified. -All requirements are clearly labelled (such as must have lamp)
Overview for classes	3 Marks	1.5 Marks	0 Marks		
	Each class file created has an overview written explaining its purpose. OR Each class file in the java doc has an overview written explaining its purpose.	Some of the class files created have an overview written explaining its purpose. OR Some of the class files in the java doc has an overview written explaining its purpose.	None of the class files created have an overview written explaining its purpose. OR None of the class files in the java doc has an overview written explaining its purpose.		
Algorithms	14 Marks	11 Marks	8 Marks	4 Marks	0 Marks
	Algorithms or Javadoc created for each Method inside each class (excluding setters/getters/toString) Each algorithm created clearly addresses the following: -requirements of each method clearly identified -all inputs, outputs, and processing are clearly identified OR Each method in the javadoc has a clear description which includes: -what the method does -Requirements of each method clearly identified	Most algorithms or Javadoc created for each Method inside each class (excluding setters/getters/toString) Each algorithm created clearly addresses the following: -requirements of each method clearly identified -all inputs, outputs, and processing are clearly identified OR Each method in the javadoc has a clear description which includes: -what the method does -Requirements of each method clearly identified - expected outputs, and processing are clearly identified	Approximately half the algorithms created for each Method inside each class (excluding setters/getters/toString) Each algorithm created clearly addresses the following: -requirements of each method clearly identified -all inputs, outputs, and processing are clearly identified OR Each method in the javadoc has a clear description which includes: -what the method does -Requirements of each method clearly identified - expected outputs, and processing are clearly identified	Less than half the algorithms created for each Method inside each class (excluding setters/getters/toString) Each algorithm created clearly addresses the following: -requirements of each method clearly identified -all inputs, outputs, and processing are clearly identified OR Each method in the javadoc has a clear description which includes: -what the method does -Requirements of each method clearly identified	No appropriate algorithms created. OR Each algorithm created does not clearly addresses the following: -requirements of each method clearly identified -all inputs, outputs, and processing are clearly identified OR Each method in the javadoc does not have a clear description which includes: -what the method does -Requirements of each method clearly identified - expected outputs, and

	- expected outputs, and processing are clearly identified			- expected outputs, and processing are clearly identified	processing are clearly identified
Bug report	5 Marks	3 Marks	2 Marks	0 Marks	
	At least 10 bugs identified during development, each outlining the issue, who discovered it, and a solution provided to rectify the issue	At least 7 bugs identified during development, each outlining the issue, who discovered it, and a solution provided to rectify the issue	At least 4 bugs identified during development, each outlining the issue, who discovered it, and a solution provided to rectify the issue	Less than 4 bugs identified during development.	
Allocation of Duties	2 Marks	1 Mark	0 Marks		
	All the following criteria are met: -Allocation of duties for each member of the group is clear. -Workload is evenly and distributed	One of the following criteria is met: -Allocation of duties for each member of the group is clear. -Workload is evenly and distributed	None of the following criteria are met: -Allocation of duties for each member of the group is clear. -Workload is evenly and distributed		

Marking Guide – Coding 65% of the total mark

Coding Style	5 Marks	3 Marks	0 Marks		
	All the following criteria are met: Code is indented appropriately Consistent block bracing has been used Appropriate naming conventions used for variables	Two of the following criteria are met: Code is indented appropriately Consistent block bracing has been used Appropriate naming conventions used for variables	Less than two of the following criteria are met: Code is indented appropriately Consistent block bracing has been used Appropriate naming conventions used for variables		
Commenting	5 Marks	3 Marks	2 Marks	0 Marks	
	At least 10 appropriate comments have been used in the code	At least 7 appropriate comments have been used in the code	At least 4 appropriate comments have been used in the code	Less than 4 At appropriate comments have been used in the code	
User Input	6 Marks	4 Marks	3 Marks	2 Marks	0 Marks
	A range of user-inputs demonstrated showcasing at least 3 datatypes accepted. AND All user inputs are appropriate and have appropriate data validation techniques applied	A range of user-inputs demonstrated showcasing at least 2 datatypes accepted. AND Most user inputs are appropriate and have appropriate data validation techniques applied	A range of user-inputs demonstrated showcasing at least 2 datatypes accepted. AND Approx. half the user inputs are appropriate and have appropriate data validation techniques applied	A range of user-inputs demonstrated showcasing at least 1 datatype accepted. AND Approx. half the user inputs are appropriate and have appropriate data validation techniques applied	Little or no user input demonstrated OR Approx. **less than** half the user inputs are appropriate and have appropriate data validation techniques applied
Datatypes	6 Marks	4 Marks	2 Marks	0 Marks	
	Demonstrated at least 4 appropriate data types used showing an understanding of these data types written in the code	Demonstrated at least 3 appropriate data types used showing an understanding of these data types written in the code	Demonstrated at least 2 appropriate data types used showing an understanding of these data types written in the code	Demonstrated less than 2 appropriate data types used.	

Selection Statements	8 Marks	6 Marks	3 Marks	0 Marks	
	At least 10 selection statements used. AND Appropriate use of selection statements that cover each of the following: -If-else -Switch-Case -Nesting -Implementation of user menu	At least 7 selection statements used. AND Appropriate use of selection statements that cover each of the following: -If-else -Switch-Case -Nesting -Implementation of user menu	At least 4 selection statements used. AND Appropriate use of selection statements that cover at least 3 the following: -If-else -Switch-Case -Nesting -Implementation of user menu	Less than 4 selection statements used in the code OR Use of selection statements that cover less than 3 of the following: -If-else -Switch-Case -Nesting -Implementation of user menu	
Iteration	8 Marks	6 Marks	3 Marks	0 Marks	
	At least 8 different loops used appropriately AND Appropriate selection of loops applied including the use of: -while -for -do-while	At least 6 different loops used appropriately AND Appropriate selection of loops applied including the use of: -while -for -do-while	At least 4 different loops used appropriately AND Appropriate selection of loops applied including the use of any 2: -while -for -do-while	Less than 4 different loops used appropriately OR Only one of the following loops applied: -while -for -do-while	
Random Numbers	3 Marks	2 Marks	1 Mark	0 Marks	
	At least 3 Random numbers are generated and used appropriately in the code	At least 2 Random numbers are generated and used appropriately in the code	At least 1 Random number is generated and used appropriately in the code	No Random number is generated	
Arrays	6 Marks	4 Marks	2 Marks	0 Marks	
	At least 4 different arrays created with appropriate datatypes AND Appropriate array type selected (standard/dynamic) AND Both: -Standard Arrays -Array Lists are used in the code	At least 3 different arrays created with appropriate datatypes AND Appropriate array type selected (standard/dynamic) AND Both: -Standard Arrays -Array Lists are used in the code	At least 2 different arrays created with appropriate datatypes AND Appropriate array type selected (standard/dynamic) AND One of: -Standard Arrays -Array Lists are used in the code	Less than 2 arrays created with appropriate datatypes Or Less than 2 arrays use appropriate array type.	
Methods	10 Marks	8 Marks	5 Marks	2 Marks	0 Marks
	At least 10 different appropriate methods have been created (excluding main, accessor and mutator methods) AND At least 4 different return types used	At least 7 different appropriate methods have been created (excluding accessor and mutator methods) AND At least 3 different return types used	At least 4 different appropriate methods have been created (excluding accessor and mutator methods) AND At least 2 different return types used	At least 4 different appropriate methods have been created (excluding accessor and mutator methods) OR At least 2 different return types used	Less than 4 different appropriate methods have been created (excluding accessor and mutator methods) OR Less than 2 different return types used
Classes	8 Marks	6 Marks	3 Marks	0 Marks	
	(Excluding the driver file containing the main method) At	(Excluding the driver file containing the main method) At	(Excluding the driver file containing the main method) At	(Excluding the driver file containing the main method) No	

	least 3 additional classes created. Each of these classes has at least one object instantiated.	least 2 additional classes created. Each of these classes has at least one object instantiated.	least 1 additional class created. Each of these classes has at least one object instantiated.	additional classes created.	[redacted]