ASSIGNMENT REQUIREMENT

MODULE : INTRODUCTION TO PROGRAMMING

COURSE : DICT/DNDFC

CODE : ITGP2008\V1.0G4

OVERVIEW

In this assignment student will demonstrate their competency in developing a program in a team using fundamentals of programming such as variables, conditional and iterative execution, collections (list, dictionary or tuple), functions, packages and Object Oriented Programming (OOP).

The assignment consist of induvidual and group element which will be completed by 4 – 6 students in a team. The entire assignment are divided into three parts which cover (A) program development, (B) team report and source code submission, (C) individual report.

You and your team are highly advised to carefully design the game structure and distribute the component evenly to each member. The program required collective effort and a good teamwork. Therefore, every member must contribute by designing and developing the component assigned and then integrate into the main program. Working alone/in silo may result in unfavourable outcome / marks for the module as a whole.

WEIGHTING:	**40%**
DUE DATE:	**4 Sep 2020 2100hr**

ASSIGNMENT REQUIREMENT

Project Summary: Developing a Turn-Based Battle Game

Your team has been assigned to develop a console or GUI turn-based battle game. The game allows player to setup a team of units for battle (minimum of 1 unit, default is 3). Each unit has a unique name and attributes like health point (HP), attack point (ATK), defence point (DEF), experience point (EXP) and a rank (default is level 1).

A unit can be either a Warrior, a Tanker or a Wizard whom have different strength in ATK and DEF point. In addition, a Wizard has special ability to cast spells that can impact friendly and enemy units (Heal, Poison, Cure, Freeze).

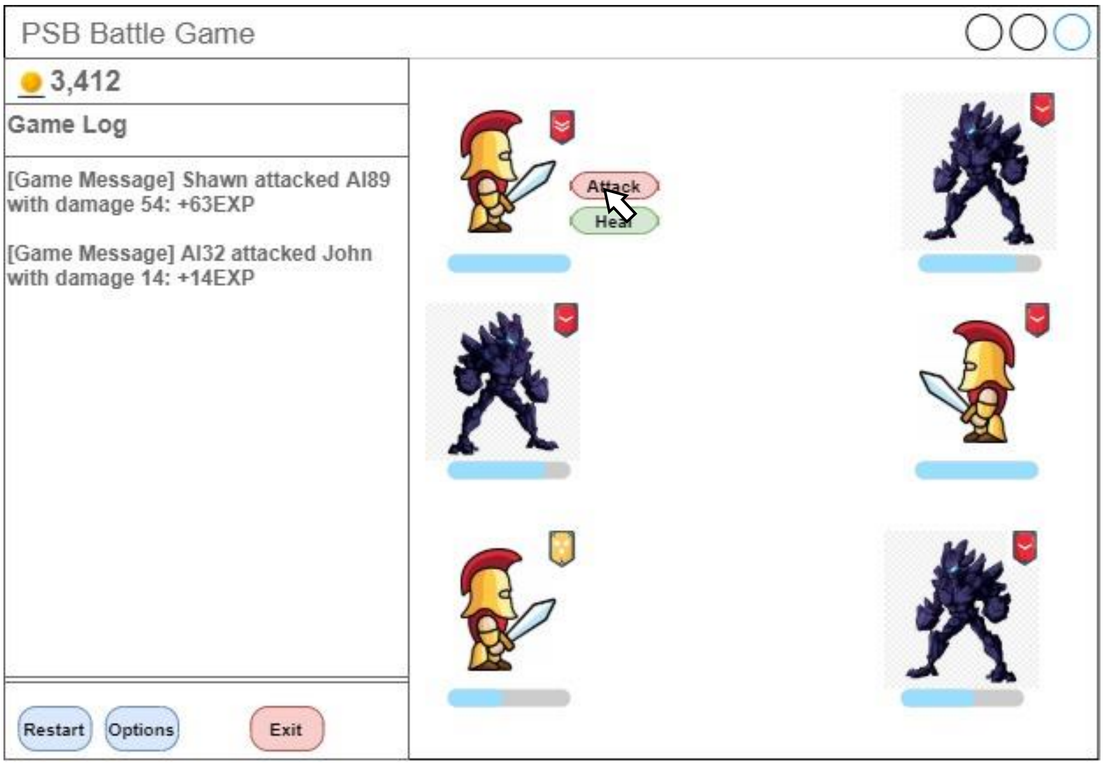
The suggested initial values for each unit's attribute point are described in requirement details under Part A – Game Setup section.

The game will automatically setup either (1) player vs Artificial Intelligence (AI) or (2) two-player mode, of which both teams are made up of same number of units.

For AI team, the type of units will be assigned randomly or by specific AI algorithm. The name of units will be specified by the player, whereas AI unit names can be defined with prefix + random numbers (eg AI87). For two-player mode, each player will be able to go through the same unit selection process either through console or GUI.

For each turn, player can select an active friendly unit (non-frozen or dead) to perform an action on a target unit. Units which are severely damaged (i.e. HP equals to or less than 0) will be considered killed (or flagged as "dead").

The game will end when one team destroys all of opponent's units.



[Game Message] Shawn attacked AI89 with damage 54: +63EXP

[Game Message] AI32 attacked John with damage 14: +14EXP

PART A – Program 32%

A-1: Program's minimum core features: [max 17%]

Features	Description
Game Setup	This feature allows player to setup a team of 3 units and select which profession (Warrior / Tanker / Wizard) each unit will be. Player will also need to assign a **<u>unique and non-blank name</u>** for each of the unit within the team. Suggested initial statistics for each unit's attribute can be found in the following matrices. Students can modify the numbers accordingly for more balance game play.

Attribute	Warrior		
	Level 1	Level 2	Level 3
Health Point (HP)	50	60	70
Attack Point (ATK)	8	9	10
Defence Point (DEF)	3	4	5

Attribute	Tanker		
	Level 1	Level 2	Level 3
Health Point (HP)	60	70	80
Attack Point (ATK)	5	6	8
Defence Point (DEF)	5	6	8

Attribute	Wizard		
	Level 1	Level 2	Level 3
Health Point (HP)	40	50	60
Attack Point (ATK)	3	4	5
Defence Point (DEF)	2	3	4
Special Ability (Cast spell)	Heal +5HP Poison -3HP each turn Cure (remove poison) Freeze (unit lose 1 turn)	Heal +8 HP Poison -5HP each turn Cure (remove poison) Freeze (unit lose 1 turn)	Heal +10HP Poison -8HP each turn Cure (remove poison) Freeze (unit lose 1 turn)

It is mandatory to implement Wizard's Heal spell. Students who implement Poison/Cure/Freeze spells will be awarded bonus (see section A-2).

Students can adjust any statistics above accordingly to balance gameplay.

All new units will have experience points (EXP) starting with zero. EXP will increase/change as per rules listed in the next section (Player Action).

Student to implement either AI or second player's team with same number of units and various unit types.

Features	Description
Player Action **(default once per turn)**	This feature allows player to select a friendly unit to perform an action (attack or cast spell), and another friendly or enemy unit as the target. The program will execute the action follow by refreshing all units' statistics like HP, ATK, DEF, rank level and whether unit is Alive, Frozen or Poisoned. For attack, the health point (HP) of enemy unit will be deducted and both attacker and targeted units will gain experience (EXP).

	Calculations and game rules for attack/spell effect as follow: • Total Damage = attacker's ATK – target's DEF + Bonus Damage (random number between -2 to 5, student can adjust range accordingly) • Target's HP deducted based on the total damage point. • Attacker's EXP increase based on the total damage point. • Target's EXP increase based on its DEF point. • A wizard unit will get 5 EXP increase for each spell casted. • A unit that is frozen will lose a full turn, after which the program should automatically un-freeze it. Eg A Warrior frozen on turn #2 will not be active in turn #3, it can only resume action on turn #4. • Heal and Cure spells should only work on friendly units, whereas Poison and Freeze will work only on opponent's units. • Poison effect will continue (deduct HP every turn) until the unit's friendly Wizard cast Cure spell on it. A unit will be promoted (level up) when EXP point reaches 50. Student should ensure this promotion limit can be easily adjusted in program. • Rank will be updated upon promotion (minimum of 3 levels). • EXP will be deducted by 50 point (or other adjusted value) after promotion. • Unit attributes (HP, ATK, DEF) will be reloaded to the new level. Gameplay must indicate the turn number during the battle (eg Turn #1 player 1's action, Turn #10 player 2's action). When all units are non-active (eg frozen), player will lose a turn automatically. A unit will be removed from the team list (or flagged as "dead") when the HP become 0 or less, and will not be available to be selected for action. This can happen right after an attack or a poison effect.
AI / Opponent Action	Once the player completed a turn, the AI team or second player will then take its turn to initiate an action. Same rules applied on the damage and experience point calculation as above. You as the game developer will need to define your own logical/creative algorithm for the AI team to make the unit and target selection.
User Interactivity	The program should allow player (user) to interact via keyboard input in python console shell using "menu" options for selection of units and respective actions. **Bonus marks will be awarded for using GUI (refer to section A-2).**
Game Message	The program should display necessary messages during/after execution of a player's action (i.e. unit names and their actions, statistics/status of all units).
Event Log	The program should record all game events with timestamp/details and save in a plain text file on the local machine.
Documentation	Include appropriate comments for all functions and important sections of codes.

A-2: Enhanced features, concept and packages applied [max 15%]

Features	Description
Game Setup & Game Play	The program can be enhanced by any of the following suggested features: • **[2 marks]** Allow user to specify maximum team size at program start (minimum team size of 1). • **[2 marks]** Allow user to save game and resume when the program is started. • **[3 marks]** Allowing user to load important statistics/configuration from external file, so the numbers can be adjusted outside of source codes where necessary to improve gameplay on-the-fly. Some examples are: o HP/ATK/DEF for unit types across all rank levels o Attack Damage bonus, Poison Damage and Heal HP o Required EXP for promotion • **[max 8 marks]** Implement spells like Poison/Cure or Freeze, effect as per section A-1. Or add any interesting spells, eg “Resurrect” to revive a dead unit, etc. • **[max 5 marks]** Each team can earn coins based on total damage points (Eg 10 points for 1 coin). The team can then recruit new units during their turn (up to max team size). Students to design cost structure of recruiting units accordingly. • **[2 marks]** Extra EXP can be gained by unit when special events happen. Eg. o Total damage point more than 10 (Attacker gain extra 20% EXP) o Target unit is killed right after an attack (Attacker gain extra 50% EXP) • **[max 8 marks]** More intelligent AI rather than based on randomized actions. • Any other interesting game play / battle rules can be applied **[marks to be awarded based on difficulty and lecturer’s assessment].**
OOP Concept	**[3 marks]** Appropriate use of Object-Oriented Programming paradigm covering the abstraction and encapsulation concept for game setup and game play.
User Interactivity	**[5 marks]** Enhance user experience with Graphical User Interface (GUI) for interaction. (Recommended packages: tkinter, pygame, turtle, PyQt, Kivy).

A-3: Technical Specification and Guidelines

You are expected to apply concepts learnt in this module when developing this game. You may use List to store the collection of units. Even better if you use Object Oriented Programming concept to manage teams and the units in the game. You may use other python libraries / packages for the development and must be included into the team’s portfolio and documentation.

PART B – Team Report and Source Code Submission 4%

The Team Report

The Team report will showcase the application development process in collective and team environment. The report indicates the completed application functionalities and also cover the technical aspect i.e. additional programming library/package/module used with proper explanation, flow chart for the process/algorithm for the main function or overall program, and also screenshot or figure for illustration.

The report will also indicate the summary of contribution from each member in completing this project.

Below are the breakdown items required for the report (report outline):

1. Cover page (with team members name)
2. Summary of program's functionalities - **it is important to highlight all core minimum and enhanced features implemented so the lecturer can test and grade accordingly!**
3. Technical sections for the following:
    a. Flow Chart indicating the main program and other important functions' process flow
    b. Additional packages / module used in developing the program. Explain how these packages / modules is installed, why it is needed and indicate which part of the programme uses these packages / modules
    c. Installation and user guide **explaining clearly how to setup and run the program**

4. Complete source code and resource files (with appropriate comments and indentation)

5. Summary of Members' Contribution (See sample provided at **Appendix B**)

PART C – Individual Report 4%

Personal Learning Reflection Report by Each Member for the Project (~300 – 800 words)

Each member within the team must produce their own individual portfolio to document their individual contribution and to demonstrate their competency in designing, implementing, test and debugging part of feature/algorithm within the program. Besides, each member must prepare a personal learning reflection report towards this assignment.

The purpose of the personal learning reflection is to allow each individual student to summarise the insights and experiences he/she has gained from this assignment / class activities within the module (Introduction to Programming) at PSB Academy. Students are required to highlight their own personal perspectives, opinions and feelings. It provides an honest summary of the work undertaken throughout the module, and the skill sets that were developed.

The key of the assessment criteria is based on how well the student demonstrated genuine engagement with the module, and how well he/she apply the Python coding skills in the Group project, or any other related class activities. The Personal Learning Reflection Report should made up of the following sections:

a. **Project Contribution**

   Member must explain and describe part of their contribution towards the project completion might it be technical or non-technical aspect. Each of the contribution must be supported by evidence i.e. photos, log book, or communication log i.e. email or any sort of communication tools.

b. **Challenges encountered and how it was overcome.**

   These are typically short summaries of moments that significantly enhanced your learning in this module. The challenges / critical incidents can be either positive or negative experiences which provided strong opportunities for your professional development. When writing about such incidents, you should reflect on the ways that they prompted new skill development, or provided enhanced understanding of course material.

c. **Sense of achievement with evidence**

   This part of reflection provides written evidence of your achievement. For example, you write about any accomplishment that made you feel proud, or any activities that you like to share as your success story.

d. **Personal statement / Conclusion**

   The Personal Statement provides an opportunity for you to summarise your newly developed skills and professional philosophies. Based on the experiences you've gained, how would you describe yourself professionally? How will you use the skills you learnt in your professional development in the future?

Tips: How to write a good reflective report:

1. **Be critical.**

   Although the content of a portfolio will be more personalised than other assignments, you should use the same level of critical analysis as you do for any essay or exam.

2. **Be comprehensive.**

   Make sure that you include a good range of experiences that exemplify your work throughout the duration of your practical group work/class activities. You might choose to highlight one or two periods of your work, but these should be contextualised within your overall experience.

3. **Don't be afraid to reveal your weaknesses.**

   Writing about your professional insecurities and weaknesses shows examiners how much you've developed throughout your course. It also enables you to reflect on theories and methods that might benefit you in future.

4. **Devise a plan for development.**

   Your Reflective Portfolio should testify to your development as a practitioner throughout the duration of your course. However, to write a really strong portfolio you should also demonstrate an action plan for future development. Think about what knowledge and skills might address the professional weaknesses that your reflections reveal, and indicate how you intend to develop these.

Mistakes to avoid in writing reflective portfolio:

The most common mistake in Reflective Writing is to be either too objective and scholarly, or too emotional and non-critical. Either mistake is equally wrong. Students should aim for a middle ground in their writing, in which they highlight their own personal feelings and reflections but analyse these with reference to the theoretical course material. Another common mistake is not providing enough relevant evidence to support your reflections. Be sure to include supporting evidence like photos, diagram, etc.

Finally, be sure to keep your portfolio well organised and professional-looking. It is true that Reflective Portfolios entail a less formal style of writing, but students sometimes believe that this allows for disorganised presentations with jumbled notes, illegible handwriting and poor grammar. Remember that this is still an academic assignment, and all the normal standards of achievement apply!

WHAT TO DO & WHERE TO START?

1. Form a team of 4 – 6 students within the same Lab Group
2. Discuss and understand the requirement of the assignment
3. Define the parts / component to be implemented as a team
4. Explore and research possible external python packages to be adopted for this project. Experiment these packages if needed and finalize solution approach as a team
5. Appoint a Team Leader
6. Develop the Program
    a. Team leader to distribute tasks within team members and set deadlines for all tasks
    b. Design, develop and test the component individually based on your assigned tasks
    c. Integrate all the source code components and test the program as a whole
7. Discuss and update lecturer regarding percentage of contribution among team members no later than Session 13 (**Team Leader to email lecturer**)
8. Prepare and submit Technical Report and complete source code to Blackboard (**Team Leader only**)
9. Prepare and submit Individual Report to Blackboard (**All team members**)
10. Submission deadline **– 4 Sep 2020, 2100hr**

WHAT NEEDS TO BE SUBMITTED?

Deliverables & Submission

1. A zip file (submitted by Team Leader)
    a. Team Report in Microsoft Word format
    b. All source code files (.py), resources and external Python packages used by the program
2. Individual Report in Microsoft Word Format

Students are to submit their project prior the due date indicated in Blackboard. Any late submission will attract mark deduction as follow:

No. of (Calendar) Days late	Mark
1	(final agreed mark) * 0.95
2	(final agreed mark) * 0.90
3	(final agreed mark) * 0.85
4	(final agreed mark) * 0.80
5	(final agreed mark) * 0.75
6	(final agreed mark) * 0.70
7	(final agreed mark) * 0.65
More than 7	Zero

PLAGARISM & PANALTY

Plagiarism is the practice of using another writer's ideas or observations and presenting them as the author's own. If students directly or indirectly use another author's words without due acknowledgements to the original source, they are guilty of plagiarism and their work cannot be accepted as academic writing.

Plagiarism is a serious offence in PSB Academy and may lead to penalties in the student's assessment, in most cases even failure of the assignment and / or module. In severe instances, plagiarism may lead to exclusion of the student from the programme of study. (*read more detail in Student Handbook on 4.12 Academic Misconduct: PAC-ADM-G02 R15 Student Handbook*)

ADDITONAL PYTHON RESOURCES

Python Tutorials:

- https://www.w3schools.com/python/default.asp
- https://www.geeksforgeeks.org/python-tutorial/?ref=lbp
- https://www.programiz.com/python-programming
- https://realpython.com/learning-paths/python3-introduction/

- GUI Overview: https://www.youtube.com/watch?v=VMP1oQOxfM0
- GUI Tutorials:
    - https://www.tutorialspoint.com/python3/python_gui_programming.htm
    - https://www.datacamp.com/community/tutorials/gui-tkinter-python
    - https://realpython.com/pygame-a-primer/
    - https://techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/

APPENDIX A - MARKING RUBRICS

Deliverables	Components	Below Expectation	Meeting Expectation	Good	Excellence
PART A Program [32 marks]	Game Setup & Game Play	0 – 2	2 – 4	4 – 5	6 – 8
	Player Action	0 – 2	2 – 4	4 – 5	6 – 8
	2-player mode / AI Action	0 – 2	2 – 3	3 – 4	4.5 - 5
	User Interactivity	0 – 2	2 – 3	3 – 4	4.5 - 5
	OOP	0 – 1	1	2	2 - 3
	Game Message / Event Log	0 – 1	2	2	2 - 3
[Part B] Team Report [4 marks]	Technical Sections	0 – 1	1	1 - 1.5	1.5 - 2
	Source code indentation, formatting and comments	0 – 1	1	1 - 1.5	1.5 - 2
[Part C] Individual Report [4 marks]	Personal Learning Reflection Report	0 – 1	1 – 2	2 – 3	3 - 4

APPENDIX B - SUMMARY OF MEMBERS' CONTRIBUTION (SAMPLE TEMPLATE)

MEMBER (% of contribution)	TASK ASSIGNED (& DESCRIPTION)	STATUS	REMARKS
Sally Foo (40%) …	Develop Main Menu	Completed	*<some comments>*
	Develop Game Setup	Completed	Developed all unit creation and setup for Player and AI
	Testing the application	Completed	All good
	………		
Michelle Lee (30%)	Develop all unit classes and their methods using OOP	Partial Complete	Use Python library xyz
	Develop GUI	Incomplete	Deciding between Pygame and Tkinter
	Preparing Team report	Completed	*<some comments>*
	………		
Johnson Lim (30%)	Develop Game Play	Partially Completed	All unit game play completed except Wizard's spell casting
	Develop a function to load unit statistics from file	Completed	Use Python library xyz
	Develop a function to load saved game from file	Completed	*<some comments>*
	…	…	…

NOTE: Each student's final score will be computed based on the following formula (max = 40 marks):

Score = Score x (contribution % / highest team member's contribution %)

Example:

If total score for PART A, B and C adds to 30 marks, based on the above sample contribution:

Sally Foo's score = 40/40 x 30 = 30 marks

Michelle Lee's score = 30/40 x 30 = 22.5 marks

Johnson Lim's score = 30/40 x 30 = 22.5 marks