**COMPENG 2SH4 Project – Statement of Contribution**

Your Group Name Bestises\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Your Name \_Leo Jiang\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Your Team Member’s Name \_Samuel Li\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**You must complete this statement of contribution without discussing it with your project partner, i.e., individually. Your statement should be concise (at most one-and-a-half page). It has three parts:**

1. Tell us about your own contribution to the development of your COMPENG 2SH4 project. For example, you can tell us about which project iterations (as mentioned in the project manual) and C++ project classes that you worked on and completed. You can provide a concise answer either in paragraph form or through bullet points.

In iteration one I took on the role of developer 2 and developed the player object, implemented the finite state machine from my PPA3, and modified project.cpp to add the movement of the snake on the screen (at this point the snake is its head).

Then after my partner implemented the body of the snake and added collision in a procedural manner. I refactored it all into being object oriented. The logic of drawing the screen and detecting the collision were moved to corresponding member functions in gameMechs. I also moved the 2d array game[][] which was a global variable before to be a member variable of gameMechs. To allow for further interactions between gameMechs and Player objects, I created a setGmechs() function in the player class such that even if the player object was instantiated first, it can still receive a reference to the gameMechs object created later. This also lead to an issue of circular references where both player and gameMechs referenced each other. This was solved by using forward declarations. I also created the base for the food object for my partner to work on.

Later, I added the speed changing mechanism based on my PPA3, and modified the UI to be more intuitive. Then I refactored the draw screen function in gameMechs to be more modular. I did so by making the adding of the snake to the board and the collision detection their own separate methods, and making the drawScreen method just be responsible for drawing the screen. I also went though the touched up on my partner’s loop logics to be more concise and readable.

At this point the project was functional and met the requirements of the base project. However the food object mainly served the same purpose as the objPos class but with another property of being eaten or not. And the logic relating to eating food was done inside the GameMechs.addBoard function, which was modified to return a value indicating the type of food that was eaten into a bool in project.cpp that implemented the grow and shrink of the snake. This was very against OOD since the add board function was only supposed to add the contents of the board to the game[][], and processing for the food should have been encapsulated inside the food object. So when implementing the bonus. I rewrote everything related to the food object, and changed it to the foodBin object which handled just about everything related to the food such as food generation, food consumption logic, and special foods. This fixed all the OOD related issues with the previous food implementation. I also changed how the length of the snake is handled. Previously it was based on the player’s score in gameMechs. It is now it’s own attribute in the player class, allowing us to change the score without having to change the snake length too. I also implemented new special foods and changed the food generation logic such that there can now be multiple foods on the board.

Finally I made tweaks to the UI of the game to be more clear and indicate to players what each special food do.

1. Repeat Part 1 above but this time tell us about your project partner’s contribution to the development of your COMPENG 2SH4 project.

My partner created the gameMechs class which was responsible for the scoring system and game condition system. He also ported most of his PPA code into project.cpp to serve as a basis. He then implemented the objPosArrayList object and used the testbench to ensure it’s function.

Then he added the logic to display the whole snake moving after I got the snake’s head to move and show in iteration 1. He also added self collision detection and the logic of game termination. And he also worked on the food object and its first implementation. He added random food generation, food collision detection, and snake growing mechanisms. He also implemented a basic special food logic but was not able to have it function properly.

Overall, my partner did the bulk of the writing of the initial implementation of most features and mechanisms of the base game, which allowed me to further refactor and refine them. While his commit messages were often not very detailed, he communicated with me in person to ensure we were always on the same page.

1. Tell us about your experience in your first collaborated software development through this project – what was working and what wasn’t. If you are a one-person team, tell us what you think may work better if you had a second collaborator working with you.

Overall it was very positive. I have worked on coding projects collaboratively before using git and I have found that clear communication between members and well thought out task allocation such that there is minimal interference between programmers is most important. And I think we did that in this project. We constantly kept each other updated on the state of the project and worked on separate features simultaneously. One area of improvement for the future would be if my partner was able to write commit messaged in more detail we will have an easier time tracking our progress through the project later on.

There was not really anything that “was not working”. However I did find that one of the hardest parts of the project was trying to understand what the code my partner wrote means. This is an inevitable part of working in a team but I think if we stuck to OOD more we would have had an easier time understanding each other’s code. I myself need to work on incremental engineering. I have the bad habit of changing a large amount of code at once. I should have aimed to create and push more intermediate stable versions to keep my code commits more understandable. I do not think my partner had a fun time reading through my one commit which added 233 lines and deleted 134 lines.