2a. Provide a written response or audio narration in your video that:

- identifies the programming language;
- identifies the purpose of your program; and
- explains what the video illustrates.

(Approximately 150 words)

My create task program is a re-creation of the classic space invaders game. I created the game using the Python language with Visual Studio Code as an editor for my programming. The purpose of my program is to let users play the space invaders game in which they control a spaceship and shoot as many incoming fireballs as possible to gain points. They also have to avoid crashing their spaceship into incoming fireballs and avoid negative points by not shooting blue fireballs. The screen-recorded video illustrates the main features of my program such as starting the game, ability to shoot bullets, incoming fireballs, current score tracking, text display, historical scores, game movements, and the result of a user crashing their spaceship into a fireball or setting a new high score.

2b. Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development.
(Approximately 200 words)

While I was developing my program independently, I faced challenges of varying magnitude. The most tricky one was tracking collisions. I created an algorithm that would detect a collision if the objects had the same coordinates, yet it could not detect all the collisions between sprites. I fixed this by creating two separate algorithms, bulletCollision and spaceshipCollision, to determine when collisions occur using the width, height, and coordinates of both sprites as parameters to create a box around the sprite. The algorithms check if these boxes overlap using inequalities, and return a true boolean if the points of one box are within the points of another box or false if the boxes do not overlap.

Another challenge was incorporating music/sound effects to align with bullet movements. I created the playBulletSoundEffect method to play the bullet sound effect using a built-in method from the Pygame module. I called this method for every

bullet displayed on the game window. This caused the program to play the sound effect for every bullet until it was not on display. To resolve this error, I called the playBulletSoundEffect method when the user pressed the spacebar so that the sound effect only played once.

2c. Capture and paste the program code segment that implements an algorithm (marked with an **oval** in **section 3** below) that is fundamental for your program to achieve its intended purpose. Your code segment must include an algorithm that integrates other algorithms and integrates mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program.
(Approximately 200 words)

I selected the gameLoop algorithm because it calls multiple algorithms, bulletCollision and spaceshipCollision. These algorithms work independently, as well as in combination with each other to help the game function. The bulletCollision and spaceshipCollision algorithms function independently since they track bullet and space collisions respectively among moving sprites using the width, height, and coordinates of these sprites as parameters, and essentially creating an invisible box around the sprite. Finally, the spaceshipCollision and bulletCollision algorithms determine whether or not a collision has occurred by checking if the boxes overlap using inequalities.
On the other hand, the bulletCollision and spaceshipCollision algorithms work in combination with one another for the program's scoring and crashing features to function. This is because both algorithms return a boolean as true if a collision occurs. If the bulletCollision algorithm returns a true, the program increases the user's score. If the spaceshipCollision algorithm returns true, the program calls the gameOverMethod and restarts the program.

2d. Capture and paste the program code segment that contains an abstraction you developed (marked with a rectangle in section 3 below). Your abstraction should integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program.
(Approximately 200 words)

I chose two classes, bulletClass, and fireballClass, as the program code segment that contains an abstraction. The classes store the attributes of the respective objects along with the functions to work with. These functions perform tasks such as controlling the movements of the bullet and fireball objects using mathematical

operators. I proceeded to call the functions within the classes for every fireball and bullet object within the display. I used bulletClass to make all of the bullet objects in my game and fireballClass to create all of the fireball objects in my game. Using these classes was beneficial in managing the complexity of my program as I did not have to manually create and delete every object in my game. Instead, I stored every fireball object in a list called fireballList, and every bullet object in a different list named bulletList. A conditional statement removes any fireball object that the user shot, as well as add more fireball objects to fireballList to be displayed on the game window. Similarly, a separate conditional statement tracks the location of each bullet object and removes the object from bulletList if it was not within the display or if it collided with a fireball.