The Create Task Submission

General Requirements

Assessment	Timing	Percentage of Total AP Score
Explore Performance Task	8 hours	16%
Create Performance Task	12 hours	24%
End-of-Course Exam	2 hours	60%

General Requirements

You are required to:

- independently develop an algorithm that integrates two or more algorithms and that is fundamental for your program to achieve its intended purpose;
- develop an abstraction that manages the complexity of your program;
- create a video that displays the running of your program and demonstrates its functionality;
- write responses to all the prompts in the performance task; and
- submit your entire program code.

Submission Requirements

Submission Requirements

1. **Video**

   Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. **Your video must not exceed 1 minute in length and must not exceed 30MB in size.**

2. **Written Responses**

   Submit one PDF file in which you respond directly to each prompt. **Clearly label your responses 2a–2d in order. Your response to all prompts combined must not exceed 750 words, exclusive of the Program Code.**

Creating Your Video

On a Mac, you can use Quicktime to record your screen as well as narrate it.

For a free online utility, you can use the Screen-O-Matic tool

https://screencast-o-matic.com/

Written Responses

2A) Written Response

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.

(Must not exceed 150 words)

Note: Your video can narrate the answers to all of the above three questions for this prompt. To be safe, you can also write your answers to those three questions.

One point for this response. (1/8)

2A) Sample Good Response

2a)

I created my game using ALICE which uses the Java programming language. The purpose of my game is for the wolf to "eat" five blue chickens before the timer runs out. The video depicts all of the chickens first changing color, then the wolf eating the blue chickens when they are clicked on. If the blue chickens run out, all of the chickens will change color again so that the goal of the game can be completed.

2B) Written Response

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. *(Must not exceed 200 words)*

Development processes are iterative and cyclical in nature and require students to reflect AND improve on what they have created. Examples of iterative development could include reflection, revision, testing and refining, and improvements based on feedback.

Incremental development refers to all of the steps leading to the creation of the application/game. (create Ball class, write bounce() method for walls, write collision method for bouncing from bricks, testing methods, correcting bounce, etc…)

Two points for this response. Row 2 and Row 3. (2/8)

2B) Points and No Points

Row 2

Do NOT award a point if any one of the following is true:

- the response does not indicate iterative development;
- refinement and revision are not connected to feedback, testing, or reflection; or
- the response only describes the development at two specific points in time.

Row 3

Response earns the point if it identifies two opportunities, or two difficulties, or one opportunity and one difficulty AND describes how each is resolved or incorporated.

Do NOT award a point if any one of the following is true:

- only one distinct difficulty or opportunity in the process is identified and described; or
- the response does not describe how the difficulties or opportunities were resolved or incorporated.

2B) Sample Good Response

2b)

I independently developed this program starting with my original method and continuing to build off of that. During the process, as I would think of an element that I needed to add to the game, I would work on the code needed for that certain part of the game. An element that gave me some trouble was the first part my wolfMove method which was meant to have the wolf move to blue chickens and eat them. I had a few problems with this method because at first, my wolf would move, but end up halfway underground. After a lot of trial and error, I was able to make the wolf move up while moving forward and make sure that when it stopped moving it was above ground. I was also presented with opportunities to add aspects to make my game better. One of these moments was when I decided that my game was too easy to play. To make the game more fun, I looked up a tutorial on how to make a countdown timer. Once I added the timer to the game, there was more pressure, which made the game realistic.

2C) Written Response

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate 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. *(Must not exceed 200 words)*

Algorithms make use of sequencing, selection or iteration.
Sequencing: Statements are executed one at a time, in order.
Selection: Use conditionals if/else if to select statements to execute.
Iteration: Use loops to iterate over statements.
3 points for this response. Rows 4, 5 and 6. (3/8)

2C) Written Response

Selected code segment implements an algorithm that includes at least two or more algorithms.

At least one of the included algorithms uses mathematical or logical concepts.

Mathematical concepts include mathematical expressions using arithmetic operators and mathematical functions. (distance between objects)

Logical concepts include Boolean algebra and compound expressions. (if, else-if blocks)

2C) Written Response Points/No Points

The algorithm being described can utilize existing language functionality, or library calls.

Response earns the point even if the algorithm was not newly developed. (i.e., a student's reimplementation of the algorithm to find the minimum value)

Do NOT award a point if any one of the following is true:

- the selected algorithm consists of a single instruction;
- the selected algorithm consists solely of library calls to existing language functionality;
- the selected algorithm does not include mathematical or logical concepts;
- the response only describes what the selected algorithm does without explaining how it does it;
- the response does not explicitly address the program's purpose;
- the code segment consisting of the selected algorithm is not included in the written responses section or is not explicitly identified in the program code section; or
- the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).

2C) Sample Good Response

You need to put an oval around the main algorithm and include it in your 2C) written response.

```
for (a, b) in users_dict.items():
    guess = str(input("Input the term that has the definition of " + b + ": "))
    if guess.lower() == a.lower():
        points_scored += 1
        streak += 1
    else:
        words_missed.append(a)
        streak = 0
    if streak >= 2:
        if streak % 2 == 1:
            print("You're on fire!! YAYY...\n")
        else:
            print("You're doing great! Keep it up!\n")
    elif streak == 1:
        print("Keep it up!! You're almost there...\n")
    else:
        print("Oh come on, I know that you can do it...\n")
```

2C) Sample Good Response

An **algorithm** in my program **is** a **for-loop** that **includes 3 sets of if**-else **statements**. The **for-loop** iterates over each item in the dictionary and **asks** the **user to enter the term which corresponds to** a given **definition**. The **1st set** of if-else-statements functions by **add**ing **1** each **to** user's **score and streak-point if** the user's **answer = key of the dictionary-item**. Else, it resets streak to 0. This demonstrates **use** of **logical concepts since if-statements use Booleans.** The **2nd set of if**-else-**statements determine which** motivational-**message to print based on** their **streak**-points. If the streak-point >= 2, it runs the algorithm's **3rd set of if**-else-**statements**, which **uses modulus**—a **mathematical concept**—**to determine which** motivational **message to display** by using modulus 2 to determine the parity of the streak-score. Additionally, if streak-point < 2, the algorithm prints out a message that tells the user to try harder. All in all, this **entire algorithm helps to achieve** intended **purpose of the program by quizzing** the **user**, and the 3 smaller algorithms (the if-else-statements) included within the algorithm determines the score and streak-points of the user **and** uses that information to **print** adequate **motivational messages**. This entire algorithm's **developed independently**.

Note the explicit use of "logical concepts" and "mathematical concepts" in the answer. Note also the discussion of how this algorithm achieves the intended purpose of the program.

2C) Another(Processing) Example

For many of you, your main algorithm is a method which calls two or more helper methods.

```
Enemy[] enemies; Player p;
void setup(){}
void draw(){
      displayEnemies();
}
void displayEnemies(){
      for(int i = 0; i < enemies.length; i++){
            Enemy current = enemies[i];
            current.drawEnemy();
            if(isHit(current))
                  enemies.remove(i);
}}
```

Main algorithm: displayEnemies

Sub algorithms: drawEnemy(), isHit(),remove() and distanceToPlayer()

Example

```
..continue from last slide
boolean isHit(Enemy e){
        float dist = distanceToPlayer(e);
        if(dist < 50)
                return true;
        else
                return false;
}
float distanceToPlayer(Enemy e){
        float dx = e.x – p.x
        float dy = e.y – p.y
        return sqrt(dx*dx+dy*dy);
}
```

Main algorithm: displayEnemies

Sub algorithms: drawEnemy(), isHit(),remove() and distanceToPlayer() (next slide)

Algorithms

- The main algorithm displayEnemies depends on drawEnemy, isHit and distanceToPlayer.
- It also uses math and logical concepts.
- It also used a loop to perform the same task over and over again without the need for redundant code.

2D) Written Response

2d. Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle** in **section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. *(Must not exceed 200 words)*

The following are examples of abstractions. PICK ONLY ONE OF THE FOLLOWING:

1) arrays/arraylists or 2D arrays of objects, e.g., arraylist of Ball objects, array of Bullet objects.

2) methods/functions with parameters. (e.g. isHit() method which accepts a Bullet object and a Asteroid object)

2 points for this response. Rows 7 and 8. (2/8)

2D) Abstractions

Two ways to meet the abstraction requirement for create task.

1) (Recommended) You should have in your program an array/arraylist of objects. (arraylist of Balls, Enemies, Bullets, etc...). This is your abstraction. You need to explain how it helps you manage the complexity of your code. Write about each of the following!
   - Write the class(template) only once, but can create hundreds of objects. (reusability , avoid redundancy)
   - If the class needs to be changed, change it in only one place. The rest of the code remains intact. (editability)
   - Helps debug. If the Enemy objects are not working, look in the Enemy class. (debuggability)
   - Use a for loop to move and display all of them together without needing to use redundant code to move them one by one.

2D) Abstraction

2) A Function/Method/Procedure with parameters can be an abstraction. For example, if your program requires a set of ball objects to collide with a set of brick objects, the method

```
boolean collide(Ball b, Brick c){}
```

can be an abstraction. Write about each of these:

- Allows you to call it repeated with different parameters(different ball/brick pairs).
- Avoid redundancy. Instead of repeating the code for every ball/brick pair, the method can be called instead.

2D) Sample Good Response

You need to put an rectangle around the abstraction and include it in your 2D) written response.

```
int numCard = 1;
for(Card c: cardList)
 {
     addObject(c,x,y);
     numCard++;
     if(numCard ==4 || numCard ==7)
     {
         x=105;
         y+= 230;
     }
     else
     {
         x+=210;
     }
 }
}
}
```

2D) Sample Good Response

I created a list consisting of 9 cards to apply abstraction. The iterative process of going through each item in the list increases code efficiency because I no longer had to add every card to the background separately (I had around 9 separate addObject statements and 9 separate calls to the card's act() method.) I made a counter variable which I use to decide what coordinates to update. For example, if the program has already added 3 cards, it must know that it needs to start the second row. I also made variables for the x and y coordinate. If the counter variable equals 4 or 7, I would need to add 230 to the y coordinate variable and reset the x coordinate variable to 105,

2D) Points/No Points

Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.

Do NOT award a point if any one of the following is true:

- the response is an ***existing*** abstraction such as variables, existing control structures, event handlers, APIs;
- the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or
- the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).

Program Code

3. **Program Code**

Capture and paste your entire program code in this section.

- Mark with an **oval** the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and/or logical concepts.
- Mark with a **rectangle** the segment of program code that represents an abstraction you developed.
- Include comments or acknowledgments for program code that has been written by someone else.

Some More Advice

Consider doing the following to make your code more readable for the AP reader.

1) Indent nicely.
2) Refactor your code. If the draw method(or your main algorithm method) is too long, consider moving some blocks of code to different methods.

Some More Advice

3) Some of you borrow some code from previous labs we did in class. For example, the Grid and Cell class or the ArrayList Lab. Address this in your written response to indicate that these segments of code/ classes are from a lab done in class. Many people from both classes use these labs. Not properly documenting this will risk getting a zero! You can say that you wrote those classes yourself.

4) If you used some code found online(a tutorial or youtube video), cite and document these references!

5) Consult the attached Scoring Guidelines to make sure your writing answers satisfy all of the criteria they are looking for!