CSP Create Task (2023) **Name:**

Below is each of the writing prompts and their sections for the AP Principles Report. Follow the directions, answers the prompts CONCISELY.

(3a) Provide a written response that does all three of the following:

Purpose: The problem being solved or creative interest being pursued through the program.
Functionality: The behavior of the program during execution and is often described by how a user interacts with it.

BAD RESPONSES:

- Purpose is to meet the requirements of the exam.
- Wrote the same thing for the purpose and the function.

*** Approx. 150 words TOTAL for all three sections below ***

i) Describe the overall purpose of the program (short, straight to the point):

The purpose of this program is practicing typing skills.

ii) Describe what the functionality of the program is demonstrated in the video.

- **Logical, step-by-step of what is occurring in your CODE during the video.**

The game window contains of several elements: first one is "main label" which is located on top of the window this label represents the name of the program. Then we can notice typing panel which contains the typing text for the user. Also window has "speed label" and "accuracy label", these labels indicate the users speed and accuracy. Speed and accuracy are calculated after user finish typing all text. At last, the window has an exit button that kills entire program.

iii) **Describe the input and output of the program demonstrated in the video.**

- **Describe what the input/output represent in your CODE.**

When user types letter the program checks if the given character matches with correct character. If user's typed character is correct the color of this character changes to green and page is redrawn. If user's typed character is not correct the program does not change color but counts as a mistake to calculate accuracy in the future.

(3b) Capture and paste two program code segments you developed that contain a list (or other collection) being used to MANAGE COMPLEXITY in your program.

***Manage Complexity*:** **You couldn't do it without a list. Meaning, you couldn't have used 20 variables instead because there could be 30, not because you're too lazy or its bad to use 20 variables.**

BAD RESPONSES:

- Use a list/array out of convience, such as instead of having 4 variables we just make an array, however the order or position relative to one-another is never needed by the program.

*** Approx. 200 words TOTAL for all three sections below ***

i) Copy/paste an image showing HOW data have been stored in the list.

```
String[] words = new String[]
        {
        "apple", "banana", "cherry", "date", "elderberry",
        "fig", "grape", "kiwi", "lemon", "mango",
        "orange", "pear", "quince", "raspberry", "strawberry",
        "watermelon", "avocado", "blueberry", "coconut", "dragonfruit",
        "guava", "jackfruit", "kiwifruit", "lychee", "mangosteen",
        "nectarine", "papaya", "passion fruit", "pineapple", "pomegranate",
        "apricot", "blackberry", "cantaloupe", "cucumber", "durian",
        "grapefruit", "lime", "longan", "lychee", "melon",
        "mulberry", "olive", "peach", "plum", "pumpkin",
        "rambutan", "starfruit", "tangerine", "tomato", "ugli fruit",
        "zucchini", "artichoke", "asparagus", "beetroot", "bell pepper",
        "broccoli", "cabbage", "carrot", "cauliflower", "celery",
        "chard", "chili pepper", "corn", "cucumber", "eggplant",
        "garlic", "ginger", "kale", "leek", "lettuce",
        "mushroom", "onion", "parsley", "pea", "potato",
        "radish", "spinach", "squash", "sweet potato", "tomato",
        "turnip", "watercress", "watermelon", "zucchini", "almond",
        "cashew", "chestnut", "coconut", "hazelnut", "macadamia nut",
        "peanut", "pecan", "pine nut", "pistachio", "walnut"
        };
```

ii) **Copy/paste an image showing the data in the SAME LIST being USED, such as creating new data from the data in the list or accessing multiple elements in the list, as part of fulfilling the PROGRAM'S PURPOSE.**

```
private String getRandomWord()
{
    String[] words = new String[]
            {
            "apple", "banana", "cherry", "date", "elderberry",
            "fig", "grape", "kiwi", "lemon", "mango",
            "orange", "pear", "quince", "raspberry", "strawberry",
            "watermelon", "avocado", "blueberry", "coconut", "dragonfruit",
            "guava", "jackfruit", "kiwifruit", "lychee", "mangosteen",
            "nectarine", "papaya", "passion fruit", "pineapple", "pomegranate",
            "apricot", "blackberry", "cantaloupe", "cucumber", "durian",
            "grapefruit", "lime", "longan", "lychee", "melon",
            "mulberry", "olive", "peach", "plum", "pumpkin",
            "rambutan", "starfruit", "tangerine", "tomato", "ugli fruit",
            "zucchini", "artichoke", "asparagus", "beetroot", "bell pepper",
            "broccoli", "cabbage", "carrot", "cauliflower", "celery",
            "chard", "chili pepper", "corn", "cucumber", "eggplant",
            "garlic", "ginger", "kale", "leek", "lettuce",
            "mushroom", "onion", "parsley", "pea", "potato",
            "radish", "spinach", "squash", "sweet potato", "tomato",
            "turnip", "watercress", "watermelon", "zucchini", "almond",
            "cashew", "chestnut", "coconut", "hazelnut", "macadamia nut",
            "peanut", "pecan", "pine nut", "pistachio", "walnut"
            };
    return words[this.random.nextInt(95)];
}
```

iii) **Identify the NAME of the list being used above.**

The name of my array is "words".

iv) **Describe what the data in the list represents in your program (game/program terminology here).**

The "words" array contains words that are used in generating the text that user will have to type.

v) **Explain how the list MANAGES COMPLEXITY in your program by explaining WHY your program could NOT be written, or how it would be majorly DIFFERENT if you did not use a list.**
   - **Do you understand the purpose of the list, and functionality it gives you that other data types do not have… e.g., index and order of items, or position of items relative to other items.**

The "word" array is necessary to my program because typing program picks a random word from the array to generate text that user will need to type. Using an array gives me a easy access to items by entering certain index.

(3c) Capture and paste program code segments you developed that contain a student-developed procedure (i.e., METHOD) that implements an algorithm that uses SEQUENCE, SELECTION and ITERATION. This procedure must use at least one parameter that has an affects on the functionality of the procedure.

- **Meaning the value in the parameter affects which lines of code execute, and which do not, through conditions in if-statements and/or loops.**

  BAD RESPONSES:

  - The same line(s) of code execute, regardless of what the parameter value is.
  - The method is **ONLY** ever called in **ONE SCENARIO**, thus only ever have one outcome.

*** Approx. 200 words TOTAL for all three sections below ***

i) Copy/paste an image showing the procedure (method) that contains sequence, selection and iteration, and uses at least one parameter as described above.

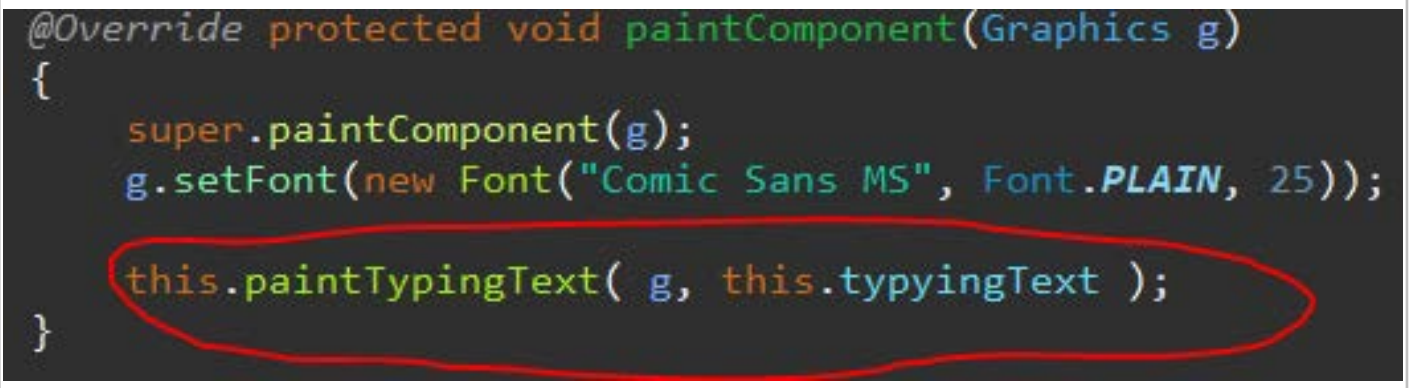
```
private void paintTypingText( Graphics g, Text text )
{
    FontMetrics fm = g.getFontMetrics();
    int text_spacing = 5;
    int text_h = fm.getHeight();
    int x_pos = 15;
    int y_pos = text_h;
    int y_spacing = 5;
    int width = this.getWidth();
    int word_width = 0;
    String word = "";

    for (int i = 0; i < text.getLength(); i++)
    {
        char character = text.getTypyingTextCharAt(i);

        if (character == ' ')
        {
            word = "";
        }
        word += character;
        word_width = fm.stringWidth(word);
        if ( text.isCharCorrect( i ) )
            g.setColor( this.correctColor );
        else
            g.setColor( this.normalColor );
        int char_w = fm.stringWidth( "" + character );
        if(x_pos + word_width >= width-190 && word.equals(" "))
        {
            x_pos = 5;
            y_pos += text_h + y_spacing;
        }
        g.drawString( "" + character, x_pos, y_pos );
        x_pos += char_w + text_spacing;
    }
}
```

ii) **Copy/paste an image showing where you CALL this procedure (method) in your program.**

- **If its multiple spots, then put together one image that shows each location.**



```
@Override protected void paintComponent(Graphics g)
{
    super.paintComponent(g);
    g.setFont(new Font("Comic Sans MS", Font.PLAIN, 25));

    this.paintTypingText( g, this.typyingText );
}
```

iii) **Describe in general WHAT the procedure does and how it contributes to the overall functionality of the program.**

- **WHAT is its purpose in the larger program, NOT what it specifically does inside the method**

Relatively to the entire program the main purpose of this function is to draw a given text on the typing panel.

iv) **Explain in detailed steps HOW the algorithm in the procedure works. Your explanation must be detailed enough for someone else to recreate it.**

- **This is EXACTLY what we/I would write on the board in ENGLISH, the step-by-step logical SENTENCES that we would then later translate to code.**

This function has a Graphics parameter which handles drawing to the window, and has a "text" parameter that is drawn on typing panel. When this function is called it initializes "fontmetrics "that allows to determine the width and height of the components drawn on the panel. This function has several variables: "text_spacing" - space between words, "text_h" - hight of the the letter drawn on the panel, "x_pos" - determines horizontal position where the character must be drawn, "y_pos" - determines vertical position where character must be drawn, "y_spacing" - spacing between rows, "width" - the width of the panel, "word_width" - width of the current word in the following loop, "word" that represent current

Then there is a loop that iterates each character of the given text string. Inside loop we have "character" variable. If this character equals to spacing, the word variable sets to empty string. Then character current character is added to word variable, and word_width is determined. Then program checks if the character matches with correct character. If it matches the graphical color sets to green, otherwise it is set to normal color which is white. Then function finds the width of the current character. Then if "x_pos" + "word_width" greater or equal to width - 190(to avoid going off the frame of the panel) and word equals to space, we move our next character to a next line, changing "x_pos" to 5; and "y_pos" plus text hight and plus y spacing. After that graphics draws a string at given x and y position. At last, "x_pos" is changed by adding "char_w" plus "text_spacing".

(3d) Provide a written response that does all three of the following.

BAD RESPONSES:

- The same lines of code execute in BOTH of your examples of calls to the method.
- In both examples, it will always be the same scenario. E.g., both locations will always be in bounds.

*** Approx. 200 words TOTAL for all three sections below ***

i) Describes TWO different CALLS to the procedure (method) identified in the previous section. Each call must pass a DIFFERENT argument(s) that causes a DIFFERENT SEGEMENT (i.e., different line(s)) of code to execute or not execute.

- **Arguments means WHAT the actual values passed where in that scenario.**

FIRST CALL:

First scenario, when function received text parameter without correct characters, and the function will draw these characters with white color.

SECOND CALL:

Second scenario, when function receives text parameter with correct characters, and the function will draw these characters with green color.

ii) **Describes what CONDITION(S) is being tested by each call to the procedure**
   - **I.e., if-statements and/or loops**

FIRST CALL:

Text parameter is an object Text. It has method "isCharCorrect(int index)" this method receives either true or false, if user's entered character matches with correct character at given index method returns true otherwise, it return false.

In "paintTypingText()" function, we have if statement which checks if the current character matches with correct character, if it does not match the character is drawn with white color.

SECOND CALL:

Text parameter is an object Text. It has method "isCharCorrect(int index)" this method receives either true or false, if user's entered character matches with correct character at given index method returns true otherwise, it return false.

In "paintTypingText()" function, we have if statement which checks if the current character matches with correct character, if it matches the character is drawn with green color.

iii) **Identify the RESULT of each call from previous section.**

- **I.e., what line(s) of code execute, and how this affects the rest of the program/game, including what is returned and its meaning, if it has a return.**

FIRST CALL:

In case first scenario occurs, character is drawn with white color.

SECOND CALL:

In case second scenario occurs, character is drawn with green color.