Create — Applications from Ideas
Written Response Submission Template

Submission Requirements

2. Written Responses

Submit one PDF document 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.**

Program Purpose and Development

2a. Identify the programming language and identify the purpose of your program. Explain your video using one of the following:

- A written summary
- of what the video illustrates OR
- An audio narration in your video. If you choose this option, your response to the written summary should read, "The explanation is located in the video."

(Approximately 150 words)

Insert response for 2a in the text box below.

I created a typing game using HTML, CSS, and JavaScript. The purpose of my program is to have the user type as much of a given text as they can in a given amount of time in order to get their words per minute (WPM) calculation. The video will present the front-end of the code consisting of the text display, the user input box, the timer, and the WPM. Additionally, it will show options that the user can change such as the amount of time given to type and the text.

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; the second could refer to either collaborative or independent program development. *(Approximately 200 words)*

Insert response for 2b in the text box below.

One difficulty I experienced, was getting the valid function working. The valid function compares the first word of the given text to the user's inputted text. When they match, the valid function would report true and false if they differ. When true, the word will then be deleted and the user will continue making their way through the passage. The text given however, is a string. I had to turn the string into an array by splitting it by spaces. I would then compare the first element of the array to the user's input when the spacebar is pressed. When the valid function is true, the first element of the array is deleted and the array must be converted back into a string so it can be displayed back into the text box. This part of the program took a lot of time to create because I had to constantly debug, using alerts to see what is going on in the backend. Another issue I had while coding was connecting all the functions together. I decided to work on my program step by step, by working on separate functions first. I had all these function that worked separately but not together. Because I was working independently, I had to self-learn the keyboard event keycode property to connect the functions together. I eventually made the function valid run when the spacebar is clicked. I also had to connect the timer with the user's input. When the user presses a key in the input box, the timer will start counting down. This was later solved by having the timer function run when a key is pressed inside the input box using the onkeypress event.

2c. Capture and paste an image or images of your program code segment that implements the most complex algorithm you wrote. (marked with a **color border** below)

```
function clock(){
  if (activated === false){
  activated = true;
  Sec30.disabled = true;
  Sec60.disabled = true;
  Sec90.disabled = true;
  text0.disabled = true;
  text1.disabled = true;
  text2.disabled = true;
  text3.disabled = true;
  text4.disabled = true;
  text5.disabled = true;
  time = setInterval(function()
  {
    startTime--;
    timeLeft.innerHTML = "Time left: " + startTime;

    if(startTime <= 0)
    {
      clearInterval(time);
      userInput.disabled = true;

      WPMCalcuation();
    }

  },1000);
  }
}
```

Your algorithm should integrate several mathematical and logical concepts.
Describe the mathematical and logical concepts used to develop the algorithm.
Explain the complexity of the algorithm and how it functions in the program.
(Approximately 200 words)

Insert text response for 2c in the plain box below.

The algorithm clock() shown above handles the amount of time the user has left to type the text they are given. When the user starts typing, the timer will start counting down all the way to zero. When the timer reaches zero, the clock must be stopped so the timer won't display negative values. Next, when the timer is zero the input must be disabled because the game is over and the player cannot type anymore. Lastly, the back-end of the program will calculate the user's words per minute only after the timer had reached zero.

The algorithm WPMCalcuation() is also implemented in the algorithm clock(). The algorithm WPMCalcuation() is called only after the user is finished typing, when the timer reaches zero. WPMCalcuation() calculates all the characters of the words that have been typed correctly and divides them by six (the amount of characters that is considered a word), which is then divided by the time in minutes. Since the time is in seconds, the time must be divided by 60. This algorithm follows the formula in calculating words per minute. Lastly, the algorithm updates the front-end so the user can see their words per minute.

2d. Capture and paste an image or images of the program code segment that contains an abstraction you developed (marked with a matching **blue color border** below)

```
function continueOn()
{
  if (valid()===true)
  {
    deleteFirst();
    userInput.value ="";
  }
}
```

Your abstraction should integrate mathematical and logical concepts.
Explain how your abstraction helped manage the complexity of your program.
(Approximately 200 words)

Insert text response for 2d in the plain box below.

The algorithm valid() shown below is another integrated algorithm that is included in the continueOn() algorithm. The valid() function compares the first word of the given text to the user's input. If they match the function reports true. In the algorithm continueOn(), when valid() reports true, the algorithm continueOn() will call another function deleteFirst(), which will delete the first element of the array and reset the user's input into blank. These integrated algorithms are important because they make the typing game function correctly.

```
function valid()
{
    firstWord = sentence[0];

    if(firstWord==userInput.value)
    {
     correctWords();
     return true;
    }
    else
   {
     return false;
    }
}
```