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.

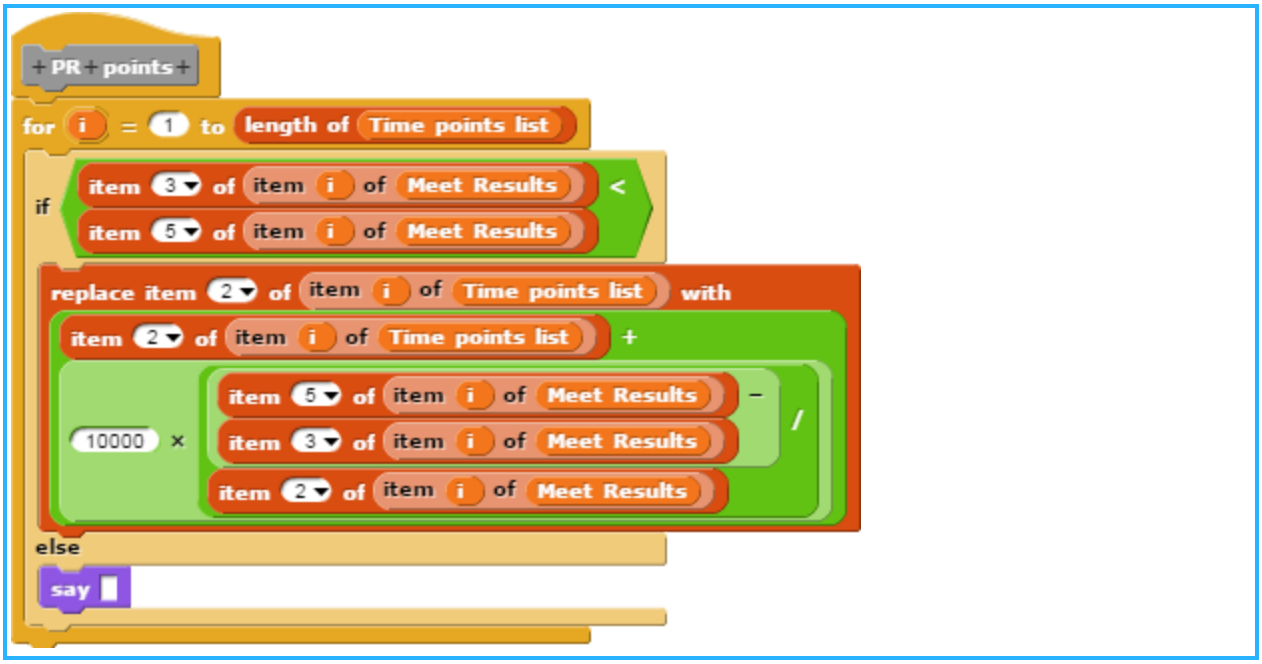
This program was written with Snap!, an online program developed by the University of California at Berkeley. This program takes into consideration a runner’s race distance, time, place, and difference in time between a new personal record and old personal record to assign points that can be used for a fantasty track league. The video shows the process an administrator would go through submitting the meet results along with a runner’s personal record, before having the program calculate those points.

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.

The most difficult part of this process was figuring out how to create a new list from an initial list, even at the very beginning creating a list from the initial meet results with names as item 1 and transferring it to a list with points that also had names as item one proved difficult. After much discussion with one another and much experimentation with list, how to place an item from one list to another was figured out. After this, the big problem was trying to figure out how to take items from list one, run them through equations, and add them together as item 2's on the second list had to be figured out. Independently I eventually figured out that using "replace" and putting in the value of the list and the newly calculated points worked to achieve this.

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)

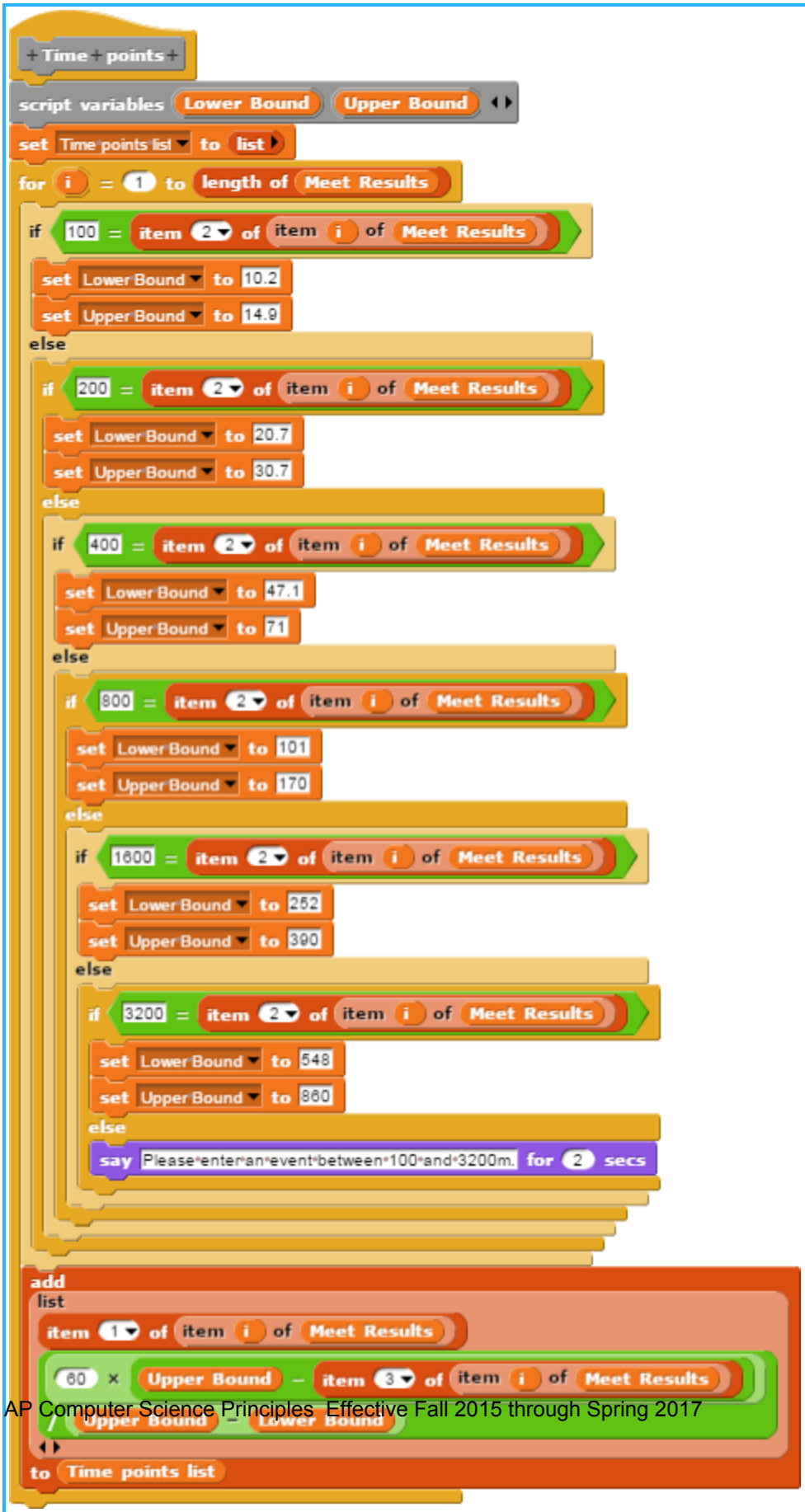


Your algorithm should integrate several mathematical and logical concepts.
Describe the mathematical and logical concepts used to develop the algorithm.
Explain the complexity of the algorithm and how it functions in the program.
(Approximately 200 words)

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

This part of the program uses a for i block, which runs through every single item in the list provided so that each individual player gets a score. Since item two of the new list already exist and already has points, the replace function is used to take the currently existing points (item 2 of item i of Time points list) and add the new points (10000X(Old personal record-New Personal record)/ Distance ran), so that the points earned from running a new personal record are accounted for. The seconds are divided by the distance ran due to the fact that more seconds are likely to be improved by the longer the race, and multiplied by 10,000 to make the points comparable to the points earned from the time and place ran. This algorithm is linear, the amount of time it takes to compute increases the same amount per item added to the meet results list.

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)


+Time+points+
script variables Lower Bound Upper Bound
set Time points list to list
for i = 1 to length of Meet Results
if 100 = item 2 of item i of Meet Results
set Lower Bound to 10.2
set Upper Bound to 14.9
else
if 200 = item 2 of item i of Meet Results
set Lower Bound to 20.7
set Upper Bound to 30.7
else
if 400 = item 2 of item i of Meet Results
set Lower Bound to 47.1
set Upper Bound to 71
else
if 800 = item 2 of item i of Meet Results
set Lower Bound to 101
set Upper Bound to 170
else
if 1600 = item 2 of item i of Meet Results
set Lower Bound to 252
set Upper Bound to 390
else
if 3200 = item 2 of item i of Meet Results
set Lower Bound to 548
set Upper Bound to 860
else
say Please enter an event between 100 and 3200m. for 2 secs
add
list
item 1 of item i of Meet Results
60 × Upper Bound − item 3 of item i of Meet Results
/ Upper Bound − Lower Bound
to Time points list

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.

This program contains 3 major abstracts. Since points are calculated based on three aspects-- time ran, place, and seconds set of a personal record-- each of those algorithms were abstracted into their own blocks. This allowed for each segment of the program to be programmed and diagnosed individually during development. If one of the individual parts was suffering from a problem, a single abstracted block could be attached to the meet results to see if the problem was in that part of the code instead of having to look through the code in its entirety. The abstraction shown above is used to calculate the points based off of the time the high school runner ran-- reasonable upper bounds and lower bounds for times that deserve points are set, and points are calculated by ((Upper bound-Time ran)/(Upper bound-Lower Bound)). A for i block is used to run through each item of the meet results. Item 2 of item i is the race ran, which sets the upper and lower bounds, and item 3 of item i is the time ran, which is plugged into the equation for points.