RUN IT CODE IT B/C
Southern California Trial Event

1. **DESCRIPTION:** Teams will run a given compiled program and recreate it.
   A TEAM OF UP TO: 2 **EVENT TIME:** 50 minutes
2. **EVENT PARAMETERS:**
   a. Teams may not bring any materials or resources.
   b. Event Supervisors will provide one internet-connected computer per team and the compiled program to be recreated. Event Supervisors are encouraged to provide access to the compiled program only available through a web terminal.
   c. Participants will be allowed to access the internet and may use any code publicly available online.
   d. Accepted programming languages in which participants can program will be announced by the Event Supervisor no later than two month before the tournament. Event Supervisors are highly encouraged to accept the latest versions of C++, Java, and Python 3.
3. **THE COMPETITION:**
   a. The Event Supervisor will provide access to the compiled program at the beginning of the 50 minute period. The compiled program will have multiple functionalities and will accept input and display the corresponding output. Teams must not have access to the source code of the compiled program.
   b. Participants will have the 50 minutes to run the compiled program any number of times and use an online IDE (such as repl.it) to write a single program in one file to recreate the compiled program.
   c. At the end of the 50 minute period, teams will submit their program as source code (.cpp, .java, .py, etc.).
   d. All input and output to the program must be through the standard streams (stdin and stdout, respectively).
4. **SCORING:**
   a. High score wins.
   b. All teams must be scored using an identical scoring rubric, containing the following weighted sections:
      i. Functionality (90%): Each functionality of the program will have a predetermined point value, which will be the same for all teams but will not be announced to participants at any point.
      ii. Documentation and Style (5%): The program should include documentation for all major functions, variables, and non-trivial algorithms and have a consistent style with proper formatting, indentation, and other white space to aid readability.
      iii. Modularity (5%): The program should use good modular design, separating the functionality of the program into coherent and reusable functions, classes, or objects.
   c. Tie Breakers: 1st - Style and Documentation Score; 2nd - Modularity Score; 3rd - Selected functionalities
5. **SAMPLE FUNCTIONALITIES**: (Click here to see example)
   a. Ask the user for their name and greet them by it.
   b. Print a pyramid of a specified size.
   c. Input two numbers and print their sum and product.

RECOMMENDED RESOURCES:
1. https://www.cplusplus.com/
2. https://docs.oracle.com/javase/tutorial/
3. https://www.python.org/
4. https://www.replit.com/