This is to implement a simple typing lesson game. Computer will generate a random string of 7 characters and you need to reproduce it within 7 seconds. You will lose points significantly if you fail to produce the same string within the set time interval.   
  
You have 1000 points in the beginning of the game. Given 7 randomly generated letters being mixed with lower and upper cases, you need to type them correctly within 7 seconds (7000 msec). You will get 500 points every time you produce the matching string within the interval. You will loose an amount of delayed time in millisecond, e.g., your typing ended in 7350 msec, so you would loose 350 points.   
  
If you misspell, you will be penalized by the total offset of mistaken letters. You will loose this offset from your points. The offset is computed by accumulating the absolute distance between two characters in the same position, one from the generated string and another from the input. For instance, the offset of "Game" and "Mag" is 81. The shorter string is padded with space(s). Therefore, |G - M| = 6, |a - a| = 0, |m - g| = 6, |e - (space)| = 69. Use ASCII table (

[http://www.asciitable.com](https://www.freelancer.com/users/l.php?url=http:%2F%2Fwww.asciitable.com&sig=0b1d7fae7dedb94d55360bc1006a3dd895089158e166a306a7f39ba664519e0d" \t "_blank)

) to see the assigned number to each alphabet under "char" and "html". If you misspell and overtime, you will be penalized the double score of the offset, plus delayed amount of time. The game ends if you reach score 5000 or 0.   
  
With 20% of chance of each character generation, a resulting string may contain wild letters, [0-9] and [%-&], which respectively indicate any number letter and any symbol letter, that means that either of the two wild letters is chosen with the probability of 0.2 when generating a random letter. Symbol excludes digits and upper/lowercase letters. Typing error is treated as an offset from '0' and '\*' respectively. For instance, the offset of Twv[0-9]JG[%-&] and TwvxJG2 is 80.   
  
There were a few ways to approximate elapsed time. The new C++ 2011 standard has a good way to do it by including <chrono> library. See the item 17 of this post (

[http://stackoverflow.com/questions/2808398/easily-measure-elapsed-time](https://www.freelancer.com/users/l.php?url=http:%2F%2Fstackoverflow.com%2Fquestions%2F2808398%2Feasily-measure-elapsed-time&sig=21d3d0e9cd4791cbf1e65e6f8df3c56bcb30d7cd1ebf241fec1e97888f6f6d5e" \t "_blank)

) to adapt into your logic. Furthermore, to activate C++ 2011 standard, you need to instruct g++ compiler with -std=c++11 option. Refer to this post #143 (

[http://stackoverflow.com/questions/17457069/enabling-c11-in-eclipse-juno-kepler-luna-cdt](https://www.freelancer.com/users/l.php?url=http:%2F%2Fstackoverflow.com%2Fquestions%2F17457069%2Fenabling-c11-in-eclipse-juno-kepler-luna-cdt&sig=e8bcdb7d0b6fbd68ee246805472886b7293d90475a4f3f4b2347304b9c738fa9" \t "_blank)

) to realize it. See the next sample game session:   
  
Your current points 1000, just type -> PEq[0-9]R[%-&][0-9]: PEq0R%0   
4958 milliseconds, you made it within the interval of 7000...   
  
Your current points 1500, just type -> nELJSgd: nELJSgd   
4297 milliseconds, you made it within the interval of 7000...   
  
Your current points 2000, just type -> gP[0-9][0-9]uCO: gP00uCO   
5816 milliseconds, you made it within the interval of 7000...   
  
Your current points 2500, just type -> [0-9][0-9]nQp[0-9]m: 00nQp0m   
6131 milliseconds, you made it within the interval of 7000...   
  
Your current points 3000, just type -> UIqIaBX: UIqIaBX   
5861 milliseconds, you made it within the interval of 7000...   
  
Your current points 3500, just type -> [%-&]ErooJ[%-&]: %-&ErooJx   
8966 milliseconds, you \*failed\* it within the interval of 7000...   
String offset is 381, your total penalty is 2728...   
  
Your current points 772, just type -> [%-&]pXwMFm: %pXmS2g   
10698 milliseconds, you \*failed\* it within the interval of 7000...   
String offset is 42, your total penalty is 3782...   
  
Bye... 