# **FAQs**

## If you are unable to view the questions clearly

Refresh the screen

## If you are unable to view the code editor

Refresh the screen

## Will refreshing the page log you out?

No

## What is wrong answer?

Your program ran successfully but gave an incorrect answer. Most probably your program contains a bug, or you are not interpreting the problem text correctly. If your code shows that it has passed the sample input correctly, then remember that there are multiple input and output cases for which your code is supposed to be evaluated, which is why in spite of passing the sample input cases, you’re still getting a wrong answer.

## I keep getting various types of runtime errors. What am I missing?

A runtime error means that the program was compiled successfully, but it exited with a runtime error or crashed. You will receive an additional error message, which is most commonly one of the following:

* 1. SIGSEGV: This is the most common error, i.e., a "segmentation fault". This may be caused e.g. by an out-of-scope array index causing a buffer overflow, an incorrectly initialized pointer, etc. This signal is generated when a program tries to read or write outside the memory that is allocated

for it, or to write memory that can only be read. For example, you’re accessing a[-1] in a language which does not support negative indices for an array.

* 1. SIGXFSZ: "output limit exceeded". Your program has printed too much data to output.
	2. SIGFPE: "floating point error". This usually occurs when you’re trying to divide a number by 0, or trying to take the square root of a negative number.
	3. SIGABRT: These are raised by the program itself. This happens when the judge aborts your program in the middle of execution. Due to insufficient memory, this can be raised.
	4. NZEC (non-zero exit code): This message means that the program exited returning a value different from 0 to the shell. For languages such as C/C++, this probably means you forgot to add "return 0" at the end of the program. It could happen if your program threw an exception which was not caught. Trying to allocate too much memory in a vector. For interpreted languages like Python, NZEC will usually mean that your program either crashed or raised an uncaught exception. Some of the reasons being in such cases would be the runtime errors mentioned above. Or, for instance usage of an external library which is causing some error, or not being used by the judge.
	5. MLE (Memory Limit Exceeded): This error means that your program tried to allocate memory beyond the memory limit indicated. This can occur if you declare a very large array, or if a data structure in your program becomes too large.
	6. OTHER: This type of error is sometimes generated if you use too much memory. Check for arrays that are too large, or other elements that could grow to a size too large to fit in memory. It can

also be sometimes be generated for similar reasons to the SIGSEGV error.

## Some ways to avoid runtime errors include the following:

* Ensure that you aren't using variables that haven't been initialized. These may be set to 0 on your computer, but aren't guaranteed to be on the judge.
* Check every single occurrence of accessing an array element and see if it could possibly be out of bounds. Ensure you aren't declaring too much memory. 64 MB is guaranteed, but having an array

of size [100000][100000] will never work.

* Ensure you aren't declaring too much stack memory. Any large arrays should be declared globally, outside of any functions - putting an array of 100000 ints inside a function probably won't work.

## What is the difference between Compile and Submit?

Every problem page has particularly two buttons:

* Compile and Run
* Submit

When you click Compile and Run, the code that you have submitted is tested against the Sample Input and Sample Output. This is for your convenience to check whether the code is working properly for the sample test cases or not.

When you click Submit, the code is checked against the internal test cases. These test cases are complicated and take your program through rigorous checks so that only the perfect code can pass the all the test cases.

1. **Having difficulty understanding how IDE works**! [https://www.hackerearth.com/docs/wiki/developers/solution-guide/](%20https%3A//www.hackerearth.com/docs/wiki/developers/solution-guide/)