There are three entities.

1) Sellers

2) Auction House

3) Bidders

Sellers:

* Sellers want to sell their items through Auction House. They would like to receive some minimum amount for a given item. This minimum amount can vary from item to item.

Auction house:

* Sellers can approach Auction House with the set of items and their corresponding minimum price. Auction takes some commission, say X% of every item sold and remaining 100-X % is given to the seller.
* The bid of any item starts with some minimum threshold which may defer from item to item.
* To deter bidding by a small margin, the bidder must increase the bid by at least Y% of the threshold at which the bidding began. Without this, the bid is not considered valid by the Auction House.
* The bidding on any item concludes after some rounds, say R or if there is no bid in a given round.
* If possible, allow an option for seller to add the item to the auction list while the auction of other items is still ongoing.

Bidder:

* Every bidder has some budget B (this could be different for different bidder).
* A bidder may or may not be willing to bid on the item being sold at the moment. Also, every bidder will have an upper limit (different for every bidder-item pair) to bid for an item beyond which they do not want to bid.

* Please provide logging for all the events inside the simulation. Such as item being sold, successive bids (successful or failed) and the verdict (item was sold to who). At the end provide **item wise break up about how much money Auction house made and corresponding sellers made.**
* There will be no external input. The entire simulation should be self-contained. Print the logs to the standard output. If possible, design a logger object that can potentially be configured to send the output to any stream, but the default configuration to be STDOUT.

1. Provide a full low-level design (every class, every member variable and every member method, all other functions) with sufficient in-source comments for a program to simulate the auctioning process as described above. Any global assumption that you make has to be clearly written in a comment at the top of the program. Any local assumptions/preconditions/postconditions may be provided immediately above a method definition. Use 'old\_' if you need to refer to old values in a post condition formula. (13 points)
2. Provide implementation for the program that simulates the process described above. (9 points)
3. Provide/describe testing strategy for your program/design. If you do not have implementation, even then provide/describe how will you test and what will you test. Obviously, writing actual unit tests or test driver would fetch more marks. (6 points)
4. Provide a make file for build and automated testing. (2 points)

General instructions:

=================

* You will be evaluated on parameters defined above which subsumes readability of your code, clarity and flexibility of your design.
* This is an open-ended question. Please do not look for a 'right answer'. Just think about how you would design such a system and simulate it.
* If you feel some information is missing, please make reasonable assumption and document these assumption in comments as instructed above. No clarification will be provided during the exam.
* You are free to look up reference manuals online for Make, googletest, C++ but you are not allowed to take/receive or provide/give any help from a person directly or indirectly.
* In some of the companies this is how a technical/coding/design interview is taken with such open-ended description. If you attempt this exam with honesty, it will help you find your own weaknesses and strength and rectify your shortcomings before you go out in the real world.
* DO NOT use multi-threading as you may get stuck in debugging some notorious concurrency related bug.

Submission Guidelines

=================

* For the source you should have only 3 files. (1) auction.h (2) auction.cpp and (3) main.cpp
* For testing you can have as many test driver files as you want named test1.cpp, test2.cpp
* A Makefile with at least two targets. 'make auction' should build the executable for simulation. There will NOT be any external input and therefore entire program should be a self contained simulation.
* All the simulation parameters should be configurable and in one place in your source (perhaps as macros in the header or as a configuration object). Parameters should be set, such that it should not take more than 2 minutes to run.
* Makefile should have a target 'make test' which should build and run all the tests. Entire testsuite should be able to run in not more than 2 minutes.
* Please assume g++-10 or higher as a compiler with -std=c++20 -Wall -Werror -pedantic -pedantic-errors flags.
* For testing, Makefile should have variable GTESTINC and GTESTLIB variables which will be configured to point to include and lib directories of latest release version of googletest. Appropriate linkage option should be provided in the Makefile.
* Only the source files and Makefile should be put in a directory, with the directory name as your ROLLNO. Compress the directory as a .zip archive (.tar or .tgz or .gz is not allowed). Name of the file should be ROLLNO.zip
* Maximum of 3 submissions is allowed on the portal, only the last one will count as the submission.
* For the purpose of this exam, FMTLIB will not be linked and should not be used.