The following assumes a student who has minimal/no prior programming contest experience before coming to CUET.

L1T1
Expected Rating-Lower Grey(below 1000)
Topics-

1. Language-Learn C upto what is generally taught in 1st semester. Then learn CPP classes and STL. For learning STL, Topcoder STL Tutorial will be helpful. For learning CPP classes, use any online resource, I learnt from learncpp.com, but use whatever you can understand from.
2. Do not focus on learning algorithms immediately. The focus is on getting an initial hands-on experience at solving problems. You may solve either LightOJ Beginner Problems Category, UVa Easy Problems, or Codeforces Easy Problems(500-800 rating)
3. Topics that should be learnt-
    a. Binary Search-and then solve problems on this category from LightOJ, as much as you can.
    b. Greedy-solve from LightOJ and CF tag
    c. Number theory-many topics will already be familiar from school, and solve LightOJ. It will be enough to learn just the simple algorithms for now.
    d. Basic Math from LightOJ
    e. Concept of Graph Theory and shortest path on unweighted graphs-Solve BFS, DFS, MST from LightOJ
4. Also solve from CF problems according to difficulty, preferably around 100-120 problems per each 100 rating range.
5. Learn how recursion works, learn how to visualize it.

Expected Number of Problems solved-250-500
Tips-
Spend around an hour or two at each problem before searching for solutions online. Not more than that, because what is important for you at this stage is to solve more and more problems, and if you spend all your day at one problem, your overall solve count will be low and you will not learn much. But make sure that you spend this time wisely-when we say we have worked for an hour, what happens in reality is that we have worked for just 20 minutes in total, and the other 40 minutes have been wasted away via other ‘useful’ procrastination. Perhaps learn to adopt the Pomodoro method.
After seeing the solution to a problem, make sure to understand the solution fully and absolutely. There should be no doubt regarding anything with the solution, and if there is, that means you have not understood it fully/internalized it. Problem solving is more about pattern matching, and being able to match an unseen problem with problems that you have seen before-but if you haven’t internalized the solution to the problems that you couldn’t solve by yourself, you have then actually just memorized it, and memorized stuff doesn’t actually stay in your mind for long-and you won’t be able to recognize patterns later as well.

Learn about different good resources as well. Geeksforgeeks, emaxx, competitive programming 3, Mahbubul Hasan Shanto vai’s book on programming contest, Shafaet Ashraf vai’s blog. These are good, but not just these-there are others. Google is your friend.

L1T2
Expected Rating- low-mid green (1200-1300)

Start learning upper-beginner concepts slowly.
Topics-

1. Start solving simple recursion problems. Being able to code recursive solutions is important.
2. Learn how to calculate time and space complexity. Realize that judge runs 1e8 operations per second roughly, but this is not strictly true, and modulo operations are more time intensive than normal addition. In modern judges, if code consists of simply addition and subtraction, 1e9 operations per second may also be possible.
3. Learn the basics of DP and memoization. Learn to visualize DP solutions and figuring out DP states from the constraints of the problem. Search for tricks on the DP thinking approach online. Learn to apply random states to DP that may actually work. Solve the easiest 40-50 problems on DP from LightOJ.
4. Learn graph algorithms like-Dijkstra, Floyd Warshall(and how they are DP problems), Bellman Ford, SCC, topsort.
5. Learn Combinatorics, and more advanced number theory topics-including multiplicative functions, i've seen them come up often. Solve them from LightOJ, and google for problems on those topics on UVa.
6. Learn how to think. Thinking is a skill that can be learnt. Google is your friend.
7. Learn the basics of segment tree, fenwick tree and prefix tree(or trie). Solve the first few problems on LightOJ.
8. The above are the essentials. You may also solve 100 problems from each rating range at Codeforces as well-preferably from 1100-1500.

Learning how to think is important at this stage. You should have gotten a feel for determining how much time you should give to each problem in order to learn properly-however, I'd say do not make it less than 30 minutes, and definitely not more than 90 minutes. But make sure that you have utilized that time of thinking fully-many of us(even me) do not utilize this time fully and do not realize it. Learn to think in a IDA* approach, that is, realizing all problems have an easy solution, and can be solved in a few steps. If you find your train of thought getting too complex, or taking too many steps/difficult to visualize, then it is likely not correct, and if it is, it is most definitely not the intended solution. Learn to abandon current train of thought as soon as you realize that it may be getting too complex for you-these problems are made for humans by humans, not for Einsteins by Einsteins. A problem being difficult usually doesn't mean that the idea behind is it actually complex/hard to understand, but more often than not, the next correct step in train of thought being tricky to determine-but the step itself being actually easy.

You should have also realized that this competitive programming business is no joke. The future world finalists are currently working very hard, and while you are watching movies/browsing the net, they are currently solving problems at their ACM Room. I myself used to spend almost the whole days thinking on problems. I used to think on problems while I was commuting from home to CUET and vice-versa. I used to think on problems when eating, bathing, even just before sleeping. I used to discuss problems with my friends as well whenever a class was cancelled.

Also, do not pick too hard problems, problems in the difficulty range where you find yourself seeking the solution almost every time. Even if you feel that you have understood the solution fully, you may not have fully internalized it-and possibly even indirectly memorize it. Ideally you must pick problems, that, if they were to appear in a contest, you can almost get the idea, but couldn't get a clear picture of the full solution within contest time.

And at the minimum, do every CF contests. Even if there is a CT the next day. And solve the problems that you couldn't solve the next day-this one step is very, very, very important. This is called upsolving, and upsolve the next one or two problems that you couldn't solve in contest time. Perhaps leave the others for now.

Please do look at the official solution of problems, even if you have managed to solve one by yourself. You may find new ideas/trick there. Make a habit of learning from other people's code too.

Reading CF editorial is also an art. Do not read the whole editorial at once. Look at it part by part-like let's say, read the first paragraph, stop and think on what the next steps could be. Can't progress further? Read the next paragraph/step then. And continue.

Maybe read code instead of editorials at first? This may or may not work, but I have seen that reverse engineering the idea of the solution from the code helps to internalize the solution better in one's mind, and if you can't understand the idea of the solution of the code, the perhaps read the editorial. Or maybe you could just skip to the editorial from the beginning if you wish.

Sometimes look at other people's code after ACing a code, you may learn new tips/tricks there.

And do hide CF tags. 50% of solving a problem is about identifying the correct category of the problem, and you'll be stunted in this respect.

Expected Number of Problems solved (including previous semester)- 500 at the very least, more like 1000 if you hope for qualifying for WF in the future.

L2T1
Expected rating-1450-1700
Crucial semester. You must start learning intermediate and advanced topics steadily.

1. Learn Probability and expected value, and different topics related to it like-expected value of probability, etc.
2. Learn extended euclid, Chinese Remainder theorem, inclusion exclusion. You must learn their proofs by heart as well.
3. KMP and Hashing, and solve problems on LightOJ and CF.
4. Lazy Propagation and persistent segment tree. Solve the GSS problems on SPOJ, they are very good.
5. Game theory, nim and sprague grundy. I liked some game theory problems on Codechef, can't remember them though.
6. Learn whatever simple algorithms that you haven't learnt, like ternary search(and ternary search on integers), RMQ Table.
7. Solve hard greedy problems, and try not to see solutions for them. They help you learn how to think. You can find those on CF tags, and judge their difficulty via rating tag.
8. Learn bridges and articulation points. Solve related problems on LightOJ.
9. Also solve CF problems parallely, choosing problems according to rating. Ideally you must solve 100 problem per each 100 rating range.

By now you must have formed stable teams. You must take take part in team contests regularly-at least one every week. Select 2 star contests from CF gym at the beginning, and slowly move on to 3 star contests. Or you may select contests from Timus as well. Discussing unsolved

problems after the contest is important too, make sure to solve them. Maintain spreadsheets containing daily log of what one has solved.
Expected number of problems solved-800 at the very least. Highly motivated people can go for 1500. You can do it.

L2T2
Expected rating-1600-1800

1. Start learning advanced topics. If you have followed everything well upto this point, probably you have achieved a high even level to be able to decide your own course of practice at this time. By now you must have definitely learnt (probably not mastered) basic topics.
2. Learn max flow, min cost max flow. Learn dinic algorithm.
3. Learn matrix expo and gaussian elimination, bipartite matching, weighted bpm.
4. Suffix array and aho corasick.
5. Euler Trail.
6. Solve all above topics from LightOJ.
7. SQRT Decomposition and Mo's, google for a list of problems on CF.

You must also start solving from CF frequently. Solve as much div2C and div2D as possible-(minimum 300 in total, around 500 if you are serious).
In team contests, select 3-4 star contests in CF Gym.
Expected number of problems solved-1000 at least, and if you want WF, 2000 minimum.
You must have solved around 350 Problems in LightOJ by now, and more than 1000+ on Codeforces.

L3T1
Expected Rating-1600 if you have not worked particularly hard. 1900+ else, and even touching 2000+ sometimes.

1. By now you must have learnt the basic and intermediate things of almost all topics. Start specializing, discussing among teammates.
2. Do not solve topic wise anymore, except the topics that has been alloted to you in your team.
3. For example, you decide to specialize in DP. Learn complex DP techniques like Convex Hull trick, Knuth Optimization, etc.
4. Or if you want to specialize in Math, learn FFT, NTT and more advanced number theory topics.
5. For data structure, splay tree, treap, HLD(solve QTREEs from SPOJ) etc.
6. For graph, more esetoric problems.
7. There must be a geometry solver in each team. I do not know much about geo actually. But by now you must have a good idea of what to do, right?
8. I mean, by now you must have a good idea of what topics you want to do, and further specialize on them. The above are the topics that must be learnt by one member of the team, but there may be more rare topics as well.

You may start solving Timus Problems. Solve them according to difficulty and start solving from the difficulty range that you feel comfortable with-I'd guess it would be anywhere from 450-600 if you did everything correctly. Solve as much as you can, preferably around 500 of them.
OR,

You may solve CF problems as well. The rating ladder on a2oj is quite good. Select the ladder which you feel comfortable with. By now it should be at least 1700. Solve up till 2200+ ladder, including the extra ones. Also focus on div2D and div2Es on cf, solve as much as you can.

CF Gym-4 stars.

Expected Number of Problems-If you just want a good rank in national contests, around 1000 for each member of team. WF? Minimum 2500.
L3T2
Expected rating-1700 if you have been lazy. 2200+, even red if you have worked hard.

Assuming you have worked hard, you have become the best contestant CUET has ever seen. Your name and fame spreads in BD competitive programming landscape.