SAMAN AMARASINGHE: Welcome to the final competition. So you guys survived this far. And I don't know how. I think, probably, with very little sleep. And knowing all the emails that I have seen and all the check-ins.-- I don't know how many people had-- Who had any sleep within last 24 hours?

AUDIENCE: Haha.

SAMAN AMARASINGHE: Oh, OK. [LAUGHTER] OK. And also what we looked at was the quality of the results keep improving because he was running tests to check what's going on. And when we moved to a machine, the things ran very fast. We're like, wait a minute. Why is this running that fast? Because people had checked in more and more, faster and faster versions. So things improved, not because we checked in faster machines, but you guys actually did a lot of work.

So first before we get started, I want to thank Akamai. So we and Akamai got this nice place. And then we are going to give a demo. And we're going to have a party sponsored by Akamai, which is really cool. So you guys actually see something-- what the real-life people who care about performance do. So it will be really fun.

And also, I also want to thank most of our MIT POSSE that help you guys a lot in meeting with you individually. So we have people from Akamai, [UNINTELLIGIBLE] Digital, Google, Lockheed Martin, MathWorks, Wordica, VMWare and couple of independent consultants who basically put lot of their sweat and edge in helping you guys.

CHARLES LEISERSON: And Microsoft.

SAMAN AMARASINGHE: And Microsoft.

CHARLES LEISERSON: So let's give a hand for the posse, MIT POSSE.

[LAUGHTER]

CHARLES LEISERSON: And a couple of the sponsors. SAMAN AMARASINGHE: So here is the process we are going to do today. Today's about the competition. So we've set up a very strict set of rules that we are going to go through, which is not going to be the way we are going to grade you guys. Because we are going to be little bit more lenient. So this is the set of rules we are going to do.

So we had a lot of these images that you get submitted. And some images were not exactly same. People did little bit things. They made them look little bit different. So what they said was instead of we're deciding. We will let you guys work on images which can participate. OK? So don't be harsh to your friends.

So what we'll do, we'll first go through the images. And then well you guys can say which images should be part of the competition. This is for the competition. Then at the end, after you get the Akamai sponsored prize-- not for the grading. So the fact that the images start to count, that doesn't mean you get zero in the grade. But you won't get to participate in the competition.

So after that, what we have done is we looked at all your projects. Ran them on our 12 core machine. And we will show you and seed them. We'll say here is the order of everything happening. And there are--

AUDIENCE: [LAUGHTER]

SAMAN AMARASINGHE: There are some-- Basically, what we'll do is we'll pick out the top set of seeds in them because we don't have time to run through everybody. And basically, a lot of these things live are not for the 48 core machine.

And then the-- accepted image that ran faster, got generated faster in the 48 core machine, wins the prize. So that doesn't mean that when you start grading, we will look at all the images and we will do some averaging. We will do a lot more fair. But hey, this is a competition. We just need to find the top guy so we have a top group. So we have a much more faster thing. SAMAN AMARASINGHE: So to get started. So here are some of the images people generated. So what we want to do is show the reference image. That's the reference image. And what we'll do is we'll first go through everybody's images generated, so you get a feel for all the variations out there. And then after that, we'll ask you to go out. We'll ask you if you don't want that image to be part of this.

[LAUGHTER]

SAMAN AMARASINGHE: You'll see. OK. So this is the reference image. Why don't you go through-- Charles. Mike.

AUDIENCE: Yep?

SAMAN AMARASINGHE: Why don't you go through the rest of the images.

AUDIENCE: OK. So before we we get into the images that we actually rendered in the last few hours, I just wanted to like honorable mention for this. I think this is the coolest. I'm sure you guys have been looking at a whole variety of broken ray tracer images But I think this is pretty much the coolest thing I've seen. So honorable mention for that.

And so now let's look at things that people actually submitted. First we're going to go through all the images and sort of look at the range. And then we'll go back and vote on them. And so I'm just going to flip through the ones on the right and give a couple seconds to each.

SAMAN AMARASINGHE: So see if they can identify the artifacts.

AUDIENCE: Which one?

CHARLES LEISERSON: You got to say when you're going because otherwise people aren't going to realize that you've been through three of them already. Here's image one.

SAMAN AMARASINGHE: Image--

AUDIENCE: Here's the next one.

SAMAN AMARASINGHE: Image two.

CHARLES LEISERSON: That's five. Four, five.

SAMAN AMARASINGHE: Image three. image four, image five, image six, seven, eight, nine, ten, 11, SAMAN AMARASINGHE: 12, 13, 14, 15, OK, 15. There are some groups that we could not get the rendering. Either it crashed, or after ten minutes it was still hadn't produced an image. So they're not a part of the competition. And of course we run them for the finals. OK, now the vote time. OK, so this image, in or out? Who thinks out? Whoa, you guys are a harsh crowd. OK.

[LAUGHTER]

CHARLES LEISERSON: Who thinks in? Who thinks in?

SAMAN AMARASINGHE: Who thinks in?

CHARLES LEISERSON: We've got to get you both. Who thinks in?

SAMAN AMARASINGHE: So we know--

AUDIENCE: Do we get screwed for this?

[LAUGHTER]

SAMAN AMARASINGHE: Imagine that. OK. That image is out. You can take notes, OK.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: OK, next image.

AUDIENCE: Number one's out.

SAMAN AMARASINGHE: OK, this one, in or out? Out. OK.

AUDIENCE: Probably the same.

SAMAN AMARASINGHE: OK, it's out.

AUDIENCE: It's got the funny ring at the bottom, which I guess that's what most of you are picking up on. I think that's kind of disqualifying for the derby.

SAMAN AMARASINGHE: OK, how about this? Out? In or out?

AUDIENCE: Out.

SAMAN AMARASINGHE: Out. Whoa.

CHARLES LEISERSON: Wow, you guys are vicious.

SAMAN AMARASINGHE: Exactly. OK, how about this? How many people think they should be out? OK, I guess that'll get in here. I guess-- OK, that image is in. OK. Make sure it's in. How about this one?

AUDIENCE: It's okay.

SAMAN AMARASINGHE: Who says it's out?

AUDIENCE: [INTERPOSING VOICES]

SAMAN AMARASINGHE: OK, Who says it's in? I think it's in.

CHARLES LEISERSON: You should look at the water.

SAMAN AMARASINGHE: Water.

AUDIENCE: The water has holes.

CHARLES LEISERSON: The water has holes. That's what's going on. The water has holes.

AUDIENCE: [INTERPOSING VOICES] It's the only one we're going to be able to reference.

SAMAN AMARASINGHE: OK, so what's the this is in charge?

CHARLES LEISERSON: Let's do it again.

SAMAN AMARASINGHE: OK. CHARLES LEISERSON: OK, you have to point out--

SAMAN AMARASINGHE: What's the problem?

CHARLES LEISERSON: --what's going on in the image.

AUDIENCE: Yeah. I think we should-- I mean, do we want to get people to like say why they think it's in or out?

CHARLES LEISERSON: It's helpful to say what's in the image, though.

AUDIENCE: Yeah.

SAMAN AMARASINGHE: OK. Let's take a vote again. How many people think it should be out?

AUDIENCE: OK, that's out. OK, yeah.

SAMAN AMARASINGHE: OK, next image.

AUDIENCE: Number 5, on the slide.

SAMAN AMARASINGHE: OK, in or out? How many people think this should be out?

AUDIENCE: Is there any problem? I mean, what's the--

[INTERPOSING VOICES]

[LAUGHTER]

SAMAN AMARASINGHE: I don't think there's a problem. OK, good. Next one. How many think this should be-- OK. That's unanimous, that's out. How many think this should be out? OK, I guess that's in. This another one?

AUDIENCE: Yeah, I think it's interesting to flip back and forth and see the difference between these two. Like, these are very close, right?

AUDIENCE: Yeah. It's obvious they're using this one.

[LAUGHTER]

SAMAN AMARASINGHE: OK, how many think they should be be out? That group once is going to kill everybody.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: How about this one? How many people think this should be out?

AUDIENCE: There's not enough caustics. You don't see the floor.

AUDIENCE: Yeah, that's an interesting point.

[INTERPOSING VOICES]

AUDIENCE: Say what?

AUDIENCE: We went past ones that look the same.

[INTERPOSING VOICES]

AUDIENCE: It's really not fair. When you have an image and then put that image right after that it's not fair.

SAMAN AMARASINGHE: We will keep this in because there's few people want it out, but it looks pretty good. OK, next image. OK, how many people think this should be-- OK, that's a clear out.

How about this image?

AUDIENCE: It looks OK. It's fine.

SAMAN AMARASINGHE: It's fine. OK, anybody want it out? No? OK, good. That's in.

Next image. How about this image?

[INTERPOSING VOICES]

[LAUGHTER]

[INTERPOSING VOICES]

CHARLES LEISERSON: We really should start the reference--

SAMAN AMARASINGHE: We need another slide to see how many people want that out.

CHARLES LEISERSON: --identical images and people could be voting.

[LAUGHTER]

AUDIENCE: Is there an explanation why all of the yellows have been so far different from the reference image?

[INTERPOSING VOICES]

[LAUGHTER]

SAMAN AMARASINGHE: So I guess this image is in. OK, next one. How about this image?

AUDIENCE: It's fine.

SAMAN AMARASINGHE: Fine. OK, anybody want it out? No? OK. How about next one?

[INTERPOSING VOICES]

SAMAN AMARASINGHE: It looks pretty good.

AUDIENCE: It's pretty good.

AUDIENCE: It's messed up on the left and the right.

SAMAN How many people-- We have to count this one. This is too close. One, two, three,

AMARASINGHE: four, five, six, seven, eight, nine, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19. 19 out. Who says it should be in?

[INTERPOSING VOICES]

SAMAN AMARASINGHE: Oh, I guess it's out.

AUDIENCE: Oh, the number of hands is more for out.

AUDIENCE: OK, and that's it.

SAMAN AMARASINGHE: That's it. So now, next thing we are going to do is, we are going to show you a rank in our seating.

AUDIENCE: Yeah, so we seeded you guys on the 12 core machines. So we seeded you guys based on the time that it took for scene three on the 12 core machines. So here is the ranking so far. Top one is number one team. And we ranked about the top 10 or so. SAMAN AMARASINGHE: Show the real run time.

AUDIENCE: I don't have the run time, actually, for the seat.

AUDIENCE: For scene one?

AUDIENCE: Yeah. Actually, I can get it real quick.

SAMAN AMARASINGHE: You had the graph, didn't you?

AUDIENCE: I don't have a graph.

SAMAN AMARASINGHE: So here are the rankings of the top ten. Do we have that graph?

AUDIENCE: What? I don't have a graph.

AUDIENCE: He has numbers.

AUDIENCE: I just have the numbers.

[INTERPOSING VOICES]

AUDIENCE: There's all kinds of other numbers, too.

[INTERPOSING VOICES]

AUDIENCE: We'll say unplug this.

SAMAN AMARASINGHE: No, nobody can see it.

CHARLES LEISERSON: It's okay.

AUDIENCE: Do we want to get the rankings?

SAMAN AMARASINGHE: No. Now, what we're going to do is we're going to start from the bottom and start running it on our 48 core machine. And while it's running, we want that team to say a little bit about what they did to get their performance. So--

CHARLES LEISERSON: We need to knock out the one--

[INTERPOSING VOICES]

SAMAN AMARASINGHE: Either come prepared when you have formed your presentation, or make your image run very slow. So you don't have to. So I guess--

CHARLES LEISERSON: First of all we have to knock out the ones whose images did not render.

SAMAN AMARASINGHE: Oh, that's true. That's true. That's true.

CHARLES LEISERSON: So maybe they all get knocked out. And then everybody goes home with an iPod.

SAMAN AMARASINGHE: Or nobody.

[LAUGHTER]

AUDIENCE: Can we just put the next one up? Here we go. Now we have time for everybody.

SAMAN AMARASINGHE: OK, there we go. So this is the time for all the images. So is there any of these images today getting knocked out, can you check that?

AUDIENCE: We're pretty close.

SAMAN AMARASINGHE: Which images get knocked out?

AUDIENCE: So I'm looking at them from bottom to top.

AUDIENCE: This is the-- I think this one's knocked out.

AUDIENCE: Number one, two, and three.

[INTERPOSING VOICES]

AUDIENCE: I think David Lamb and Victor Wang got knocked out. Because this is their image here.

SAMAN AMARASINGHE: That got knocked out. The last image got knocked out.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: What's the next group?

AUDIENCE: In between-- the A group--

[INAUDIBLE]

[INTERPOSING VOICES]

AUDIENCE: This one was the water. 19.

AUDIENCE: Yeah, that was.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: Uh-oh, they're getting knocked out.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: OK, that got knocked out, too.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: Oh, man. Yeah, nobody competing at this point.

[LAUGHTER]

[INTERPOSING VOICES]

SAMAN AMARASINGHE: That survives. That survives. Push! We have a survivor. So that might be a knockout by default.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: Oh, that is out. Oh, man.

AUDIENCE: Ours is so close though.

[LAUGHTER]

[INTERPOSING VOICES]

SAMAN AMARASINGHE: That's all right. OK, I at least we have two people right now.

AUDIENCE: Wasn't that the one that was really dark?

AUDIENCE: This is the one that was sort of darker--

[INTERPOSING VOICES]

AUDIENCE: Like they don't have enough.

[INTERPOSING VOICES]

AUDIENCE: So here's the two reference images.

SAMAN AMARASINGHE: We already voted. We can't vote again.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: No-- leave them knocked out.

AUDIENCE: Some people voted against it, but--

[INTERPOSING VOICES]

SAMAN AMARASINGHE: Yeah, but it didn't get the quorum. It didn't get a quorum to get voted out.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: How about other group? First one, let's look at that.

AUDIENCE: Who's this?

[INTERPOSING VOICES]

SAMAN AMARASINGHE: OK, that survives. That's good. Mangpo. OK. That survived, too. Two. That's all right. Li Chen.

[INTERPOSING VOICES]

SAMAN AMARASINGHE: Ooooo.

[LAUGHTER]

SAMAN AMARASINGHE: Oh, man. That's a... So that is a knockout blow there. Now what we're going to do is, we're going to run it on from 48 core machine from the bottom. So we will let other teams also, if we have time, to come and explain what they did. We are running-- who's now?

AUDIENCE: This is a Dobs.

SAMAN AMARASINGHE: Dobs. OK, why don't you guys explain what you did.

AUDIENCE: Come up here.

AUDIENCE: Need the mic?

AUDIENCE: I think the mic is off.

CHARLES LEISERSON: The mic is on, but it's only for the video.

SAMAN AMARASINGHE: Video.

CHARLES LEISERSON: You have to use the mic but you can't hear the speaking.

AUDIENCE: So we parallelized the ray tracing part. The two for loops for going through all the pixels on the screen. And then we also parallelized the main loop for caustic elimination. And we changed the-- sort of the data structure-- how that works. Used some like bounding boxes to make the intersection faster. I don't know what else we did.

AUDIENCE: We found ways to get rid of--

AUDIENCE: Mic, please.

AUDIENCE: We found ways to get rid of a lot of trigonometry, which actually helped a lot. And we got rid of the recursion in locate photons, which was a major roadblock.

SAMAN AMARASINGHE: Question.

AUDIENCE: What did you do for that? How'd you get rid of the trig?

AUDIENCE: Repeat the question.

AUDIENCE: They were converting from XYZ coordinates into spherical coordinates and then back again, over and over. So we just didn't do that.

[INTERPOSING VOICES]

[LAUGHTER]

AUDIENCE: Slow down.

SAMAN AMARASINGHE: It slow down. Uh-oh.

AUDIENCE: No. The seating was done on scene one. Right?

AUDIENCE: Oh, yeah. Oh. Wait-- Apples and oranges.

[INTERPOSING VOICES] I just ran scene three.

[INTERPOSING VOICES] 48. Scene three. So there might have been a slip.

SAMAN No. But this scene one is much more simpler. OK, good. So you're on the board

AMARASINGHE: now?

[INTERPOSING VOICES]

AUDIENCE: How is the seating being done in scene one if it doesn't have a surface that's--

[INTERPOSING VOICES]

AUDIENCE: --caustics. It might be that some people did really fast on that one and--

AUDIENCE: Because we ran everybody's scene three. And we guarantee that the top end teams are on it.

[LAUGHTER]

AUDIENCE: The results are rigged.

[LAUGHTER]

SAMAN AMARASINGHE: He detained the tape. OK, good. First team is in. Then the next team.

AUDIENCE: That's Elizabeth.

SAMAN AMARASINGHE: Elizabeth--

AUDIENCE: Scott, right?

SAMAN AMARASINGHE: There we go. Why their thing is being run.

AUDIENCE: Are you running it?

AUDIENCE: All right. So I guess for the display surface, the main optimization there was imagining two planes at the top bound and the bottom bound and seeing if a ray passed through them and then only checking that section of the grid. And then even within like that square bounding box--

AUDIENCE: Like if there's a square, and then the ray's going through like that, there's a bunch of grid squares--

AUDIENCE: Could you use the mic, please?

AUDIENCE: Like if a ray's intersecting a square, there's still a bunch of grid squares above and below it that you don't have to check. So we kind of imagined it as a-- projected it onto a 2D surface. So we didn't check additional superfluous grid squares. We also did the thing where we got rid of the trig by transforming from polar coordinates to XYZ coordinates. Although, before that, we used the fast sign and cosign approximation, which was almost as fast. Using a quadratic curve, which was kind of cool.

AUDIENCE: I think those are the main--

AUDIENCE: We did an SSE matrix multiply, which barely speeded it up, but whatever.

[LAUGHTER]

AUDIENCE: We moved stuff into...

SAMAN AMARASINGHE: So you are the team that had issue going to Intel to AMD.

[INTERPOSING VOICES]

AUDIENCE: Yeah, we were complaining about--

[INTERPOSING VOICES]

SAMAN AMARASINGHE: If you use Intel, certain instruction is not the same in AMD. OK, good. So we are waiting for the time to come?

AUDIENCE: So we ran it once, but we forgot to time it.

[LAUGHTER]

AUDIENCE: It was fast enough that we could run it again--

[INTERPOSING VOICES]

SAMAN AMARASINGHE: So by the way, I want to give a chance for the top team that get knocked out, explain what they did. Even though they are not in competition because they ran much faster in the same run. Who's the top team?

AUDIENCE: [UNINTELLIGIBLE]

SAMAN AMARASINGHE: Who's the top team up there?

[INTERPOSING VOICES]

CHARLES LEISERSON: They're getting sleep.

[LAUGHTER]

SAMAN AMARASINGHE: OK, 59. Uh-oh, now we-- Oops. OK.

AUDIENCE: Ooo...

SAMAN AMARASINGHE: Inversion off that. OK, next team.

AUDIENCE: So our team also did a same trick. That was done in the previous two teams. We got rid of the trigonometry. And also used a bounding box. Basically described in the previous team. And we also tried to use SSE. But as soon as the news about an AMD machine, we got rid of that.

[LAUGHTER]

SAMAN AMARASINGHE: Oh, my look at that speed.

AUDIENCE: Whoa.

SAMAN AMARASINGHE: Whoa. Woo!

[LAUGHTER] OK.

AUDIENCE: If you want to take-- could we take more time to talk about it--

SAMAN AMARASINGHE: Yeah, sure. Yes.

AUDIENCE: Because I'm sure you guys have some things that are different than everybody else's. In fact, I know it.

AUDIENCE: It's on. It's just on.

SAMAN AMARASINGHE: It's on. It's on. It's already recording. So for the cache I use like a three-dimensional grid. Because I realized that as the old times, the torrents is like usually one, unless there's some outlier. But I don't see one in my case. So I use a three-dimensional grid. And use a lot of locks to prevent contention. So I do the same thing for photon map. I basically implement my own photon map and just get rid of theirs. And also, the same thing, like using a grid. And I spent a lot of time debugging.

[LAUGHTER]

AUDIENCE: And we also do some minor optimization. We implement the rendering crest as a reducer. So that we-- a hyper rod object So that every time Cilk steals some other's job, you will create a local view of the rendered object. And we found there is a artifact that-- because we are seeding the rendered object. The default that seed is two, right? And it will create some artifact in our image, so we use a random number to seed that rendered object. That's what we do. And also, we do something like we change the-- no, we don't have locate photon. Yeah OK. So we have done a lot of thing to improve the original photon med but after that we decide to change our implementation of photon med so we don't use that. OK.

PROFESSOR 1: OK, good. Thank you. OK. OK, the suspense is on. So the final team. OK.

STUDENT 8: So one of the first things that happened is that we looked at the code and, like, it was all a KD tree. And we're like, what's that? So we got rid of that immediately and we turned it into something-- I think the other groups might have talked about it, but we gave it a nifty marketing name, we called it a photon cube. And so we created lots of these. We partitioned the large cube environment into many small photon cubes so we could just jump in and look in the relative air vicinity of photons that you're looking for.

STUDENT 9: Yeah. So this is like, when you do locate photon, you know the reference photon and you can create a sphere. So we can do constant time to find the boundary of the photon cube. Like go from x1 to x something, y to y and z to z. So then we don't have to traverse the tree. And we basically parallelized two part, so the one in render and another one in the trace photon that when we try to put the photon in the map.

And also when we create a photon to the cube, there is a reused vector to start a photon in each cube. But then it's not really good, because when you put the photon in the cube, you have to, like, new photon. Which means that the memory, it's like spurts throughout your program. So after that we convert that into a block of memory. And we use that when we call locate photon.

STUDENT 10: And for the last, about surfaces that intersect. There were four kinds of optimizations. First one is checking if the origin of ray is outside of the maximum bound and lower bound of surface, then and check whether by direction is outside or if it's toward the outside then we don't have to check, just return false. And second thing is like the square cube thing. When rays intersect we can make a boundary by directly mapping to the XYZ at the surface at the maximum boundary and lower boundary of the surface.

And also when you iterate through the x index, we can cut off again because a bitmap. We only have to check with the indexes which correspond to ray's z index at that cube, at that x location. And the last thing was for each square we precomputed the minimum value and maximum value of y and we compared the rays by value within that range. And we just cut off if it doesn't intersect. It doesn't overlap.

PROFESSOR 1: OK. Good. OK, now this is the suspense part. OK. We are going to run this and see. The time to beat is 18.7. OK, right? Again. OK. There we go. Let's see. How many people think it's going to beat 18.7. 18.6. OK.

AUDIENCE: 8, 9, 10--

PROFESSOR 1: It's the excitement. Keep counting. OK. There we go.

AUDIENCE: 13, 14, 15, 16, 17, 18, 19--

PROFESSOR 1: Oh, oh, oh.

AUDIENCE: 21, 22, 23, 24, 25, 26.

PROFESSOR 1: Oh, OK. It came out as 26.41. OK.

[APPLAUSE]

PROFESSOR 1: So you guys are the winners. Come up here.

AUDIENCE: Danielle.

PROFESSOR 1: Where's Danielle?

AUDIENCE: Uh, Danielle disappeared.

AUDIENCE: Yes, she did.

PROFESSOR 1: OK. Let's see whether she can find her, so OK. Just hold on a second until--

AUDIENCE: Yep, there we go.

PROFESSOR 1: You can put it down now, though.

PRESENTER: So this one's the reference.

PROFESSOR 1: Put a text meter in, that's better.

AUDIENCE: Which one's reference?

PRESENTER: This one's the reference.

PROFESSOR 1: I think they don't align. Can you align them?

PRESENTER: There we go.

PROFESSOR 1: OK, so we have a winning team.

PROFESSOR 2: We have a winning team. And so it's my pleasure to introduce Danielle Smith, who is the University relations for Akamai, and she's here to give you guys your prize. So we have three winners.

DANIELLE: I just wanted to make an entrance is why I came in. Thank you. First of all, welcome to all of you guys for coming in, and we wanted to give you these prizes. So we have some cool iPod Nanos, which you may or may not have already, but it can't hurt to have another. They kind of burn out or you lose one or someone steps on it.

[APPLAUSE]

DANIELLE: Congratulations. You guys have worked really hard.

PROFESSOR 1: Congratulations. Congratulations. Congratulations. OK. So what's next. That's it. We are done?

DANIELLE: All righty. So we have plenty of food, it's going to be right on the atrium. I know that's the most exciting part. But it should be arriving in the next 10 to 15 minutes, so we have a few minutes. I thought maybe this would go a little bit longer.

PROFESSOR 1: We do have a couple of things we can chat about.

DANIELLE: Yeah. So just a couple of housekeeping things. If you need to use the ladies' room or the men's room, it's actually right outside that door. Out the double doors, you'll see it right there. I have invited some other Akamai employees to come and mingle and say hello to you guys, so I encourage you, if you see any faces that aren't familiar, to kind of go up and talk about your projects and introduce yourself. And thanks for coming. I know we wanted to switch it up a little bit because it was the last class. We wanted to get you out of the classroom and just kind of shake it up. So I'm glad that you came to join.

PROFESSOR 1: And the knock tours.

DANIELLE: And the knock tours. Thank you. So on the back of your name tags, there's a number, one, two or three. One, two or three. So that correlates with the knock tour that you're going to be on. We have to keep it to relatively small groups. So the first tour will be at 10 past 4, the second one will be at 4:40, and the last one will be at 5:10.

PROFESSOR 1: I guess if we don't have too many people we might be able to compress some--

DANIELLE: Yes.

PROFESSOR 1: So we'll see whether we can do that, compress--

DANIELLE: Yes. So I'll just call the first tour and you guys can go down. And obviously it's optional. Don't feel-- if you're not interested, you don't have to go on the knock tour. I think that's it.

PROFESSOR 1: Good. Thank you.

[APPLAUSE]

PROFESSOR 1: We can talk about a little bit. So one thing. We are trying to get HKN feedback up to hopefully 90%, and we are right now at what, 11?

PROFESSOR 2: No, we're a little bit higher. We're now at about 16 or so.

PROFESSOR 1: 16. So that's a little more way to go from 16 to 90. I hope you guys go back and then fill out this survey. Very important.

PROFESSOR 2: Please take a few minutes to do that. It really helps us in, even if you want to say bad things, OK? It really helps overall to give feedback and the department likes to see these things. Oh yeah, thanks. Likes to see how it is that people are responding to the courses. So if you hated the course, if you liked the course, in particular, we have some questions on there that are specific to how we teach the course next year.

One of the ideas we have for next year is to introduce our recitation sections, which would focus on the use of tools. OK? So basically the conceptual stuff, much as we have the lectures now, but then also having something where it really takes you through each tool, and teaches how the use of those tools which would be taught once a week. So if you think that's a good idea, that's going to really help to have that kind of feedback. And it will certainly help the students who take it next year if we can persuade the department that that's worth staffing the sections next year.

Also if you have any positive things to say about teaching assistants, that's a great place to put it. OK? If they helped you get through this term, this is a great place to say thank you and it only takes a few minutes. OK? So I'm sure that some of the TAs put in more than a few minutes, OK, on your thing. So to take a few minutes and say thank you on that form would be-- I'm sure they would very, very much appreciate.

PROFESSOR 1: One thing, as Charles mentioned, what we are trying to do in recitation is do a little bit of programmer life skills. So I know all of you guys are probably very smart, can think about the complex algorithms and stuff like that. But there are a lot of tools that, it's not the smartness but you can spend hours trying to figure out. But if you know it, it will be useful. So what we are trying to do is have these recitations to say OK, how to use a debugger. And even how to change a makefile. Things like that.

So one thing, until the food arrives, I'd like to say is, get some feedback. Think of what is a tool you'd like to know or like to learn, or something you think is very useful for people coming next year to learn. That if you have spent a lot of time worrying and you think you spent way too much time trying to understand, it would be easier for us to think about that. I mean, I would like to get your feedback. And also some of you probably will open up with some general feedback on class.

AUDIENCE: I don't know if you want to repeat that. A more in depth thing on how to use git would be really useful. Because I know it has things to make your life easier, but I don't know how to use any of them.

AUDIENCE: Yeah. women are always breaking the branches. And it's like, yeah, we maybe know how to do this and it is really easy. And then we're like, use this git command. And it's like, now your branch is broken.

[LAUGHTER]

PROFESSOR 2: Good. So git

PROFESSOR 1: Yeah. I mean that's because those are interesting tools because, if you know everything right, you can do a lot of fun things. But if you do it wrong, you end up in huge mess that all you do is scored all over the place and you don't know. So yes. I mean I can relate to that.

AUDIENCE: If you use git correctly, if you know how to use git-- If you learn how to use git you will find yourself using it even when you're working on a one person project. git is awesome. I can highly recommend learning it.

AUDIENCE: What would be helpful is that telling us that you are running on an AMD box and AMD doesn't support SSE 4.1 instructions.

[LAUGHTER]

PROFESSOR 1: Yeah, I mean, this is supposed to be X86 compatible, all those things. But I mean, some people learn the hard way that, OK, the word is not the same even though--

PROFESSOR 2: But that is part of the lesson of the course, is how do you make high performance portable? OK? And so what you choose or don't choose to use is-- But I agree.

PROFESSOR 1: That's why we ran, we want to run it the night before because we knew things like this would happen. So we can at least give some time to say OK, look. Go change something. Were you able to fix that? OK. Good. So yes.

PROFESSOR 2: Good.

PROFESSOR 1: And sometimes the problem is, a lot of these small changes you don't know until you run into things like this and oh, somewhere in page 256 in the manual that there's a footnote say, oh, by the way. And that's what happens in there. I mean if you had listed everything that would be probably and entire encyclopedia of things that you will see.

PROFESSOR 2: So one other thing that people ask about is, what comes after this course? OK? So next term Saman is teaching the compiler course.

PROFESSOR 1: It's Martin.

PROFESSOR 2: Are you teaching with him?

PROFESSOR 1: No.

PROFESSOR 2: Oh, you're not teaching. OK. I thought you were. OK. So in any case, compiler course is a good follow-on course. For those who haven't done 6.046, that's a good course and I'll be teaching that course next term. In addition, how many people here have done a UROP? Few of you. So let me encourage you to look for UROPs. You now have some skills that will make you really valuable. And in particular my group is looking for--

[LAUGHTER]

PROFESSOR 2: Probably two more UROPs than I have already. Already Boone is one of my UROPs, started just a couple weeks ago. And so we do have some other projects in both architecture and in software-related things. And so. if you're interested, you go to my group's web page. It tells you five things you have to submit if you want to work in my group. OK? That includes things like samples of your code. Well, you ought to have that by now. And so forth. So let me encourage you to do that. And Saman also has active participation of UROPs in his group.

PROFESSOR 1: Yeah. So my employees already started doing a project in my group. And I'm also looking for a couple more people to do some compiler, runtime system, and a bunch of high performance type projects I have. And also for future course, one course I want to also mention is this IAP. There's a course that is done with collaboration with VMware to look at how to build scalable large systems using this the Spring and Java platforms.

So if they're interested in trying to see how things were running on one machine and run something that can scale up to hundreds of machines. If you have ideas of creating an I 8 with web 2.0, interesting thing. Here is an opportunity to do that. So if you're looking for interesting IAP courses, this is great.

PROFESSOR 2: I really encourage every single person who completed this class to do a UROP with somebody before you leave. It's really a way of integrating your knowledge. And it is one of the most valuable experiences that you can have at MIT is to do a UROP for a professor and get involved in the research of a group. Because in some ways it's like this course, in that it integrates knowledge from a lot of different sources. And it isn't just one thing.

But even more than this course. This course we still are dealing with artificial things. And in the researcher arena, you're dealing with real problems. So let me just once again encourage you to spend some time getting a UROP and certainly looking for AUPs and MEng. How many people here are looking at MEng as a possibility? So a fair number. Yeah, that's good. That's great.

PROFESSOR 1: I want to add to that. So these days there has been a big philosophical discussion saying why do you have to have universities? You go to OCW and sit home in your bedroom and you can follow all these courses, and you don't have to pay a mighty tuition. And wouldn't that be good enough? One part that's not good enough is this class. So I mean think about it. I mean you can't go get the OCW catalog and have this kind of a dynamic. There's no seventh floor. All in OCW. And you have from there to go and get motivated. So that's one part of that.

PROFESSOR 2: There's no John or Reed or Eric or Josh to respond quickly about what's going on.

PROFESSOR 1: That's one. But I think one other thing that a lot of people miss when they look at all of the curriculum and stuff like that is the UROP ability. So a lot of these people, that is something, it's very unique to MIT and a lot of people benefit hugely with UROPs. I have had UROPs who started with me as seniors. By the time they are graduating with their MEng they have two first out papers in the best conference out there. And everybody thinks, when they say they are looking for a job, they are finishing their Ph.D and looking for academic jobs. No. They are just finishing up their masters.

And I have a couple of students that managed to become really expert in an area by being a UROP. So they are finishing up at MIT as a generalist, saying give me a job. But you're finishing up and have some deep knowledge in some area. Sitting in and at the same level as most Ph.Ds or most really deep practicing engineers in that area. So that level, even if you don't do that, just seeing what happens in a research group, being participating on that. Pushing, seeing something to the maximum limit.

I mean in the class, that's a limit. I mean even though this class is somewhat open, says make it run as fast as possible. But most of the time in the class, it's like, we give you something, if you made that, OK, you're done with it. In research, there's no limit. You really, really push your boundaries. What's that?

PROFESSOR 2: So what do we have here, John?

JOHN: I counted in my mailbox how many messages came from 6.172 staff, and it's about 3,500 over the course of this term.

PROFESSOR 2: Wow.

[APPLAUSE]

PROFESSOR 1: How much time--

PROFESSOR 2: So, this is also, as Saman mentions, a programmer life skills course. And one of the things that you should get used to in life is thanking people who help you. OK? I think that all of the MIT POSSE members, all the people who devoted, volunteered their time to help you, could really appreciate an email that says thank you and what exactly you feel you learned from them. OK? Because it's one thing to just say thanks, so long, et cetera.

In addition, these people are contacts, part of your network of understanding what the world of software developers is all about, and can serve as useful contacts to keep. How many people use a service like LinkedIn or something to-- You probably all you mostly use Facebook. Anybody use LinkedIn at this point? So many professionals use-- I don't want to pitch for LinkedIn, but it is one of the services that lots of people are on. So what are some of the other services that people use for professional contacts?

AUDIENCE: The only thing people try to contact me on is LinkedIn.

PROFESSOR 2: Yeah. So, good idea to start connecting and building a network of people who you know and who can answer questions for you. OK? Such as, gee, I'm considering these two kinds of job offers. What should I be looking for? It might be good to get a professional software engineer's input to that. OK? Because you know these folks. You've had a few meetings with them. OK? And they know you. And they may be able to say here's my take on that, as well as getting input from your family and your friends and your professors and other people that you know.

But I do think that courtesy goes a long way in life. And they really have, these are all people, as I say, who spent time looking over your code. And you should have heard some of the comments we got from them after project one. Oh my God, they have a single letter variable names for the whole thing. Or, what are you teaching these students? And I said no, this is exactly why we want you there. OK?

So I think that if you feel in particular, whether or not you feel you got anything out of it, OK? But in particular, if you did get something out of it, it would be really good idea to not forget to send some email off to thank them.

PROFESSOR 1: OK, we want to get some feedback from them.

PROFESSOR 2: Sure.

PROFESSOR 1: Have open mic?

PROFESSOR 2: Open mic?

PROFESSOR 1: Yeah.

PROFESSOR 2: Actually, open mic, let's do it like this. Because I think it'll be easier and better for the video if we just have people come up and--

PROFESSOR 1: I don't know that they're going to be too intimidating.

PROFESSOR 2: That's OK. They can handle it. These are seasoned-- So, comments about the class.

PROFESSOR 1: How we can improve? We'd really like to hear that. I mean things you liked and didn't like. For example, these absolute gradings we did for projects had passionate support and control let's say before. I mean, did you guys feel giving the top marke for the fastest and having at least a log scale, no, we didn't do a linear scale going down. I mean, hey. Was it motivating or demotivating?

PROFESSOR 2: So what do people think was the most valuable thing out of the course? I've had dinner with a few of you, and you were very opinionated over dinner. So I don't mind hearing those comments again, but making them for other people. So I know it's intimidating, but what I want to do is get a line. As one person's talking I want other people to line up behind. And just give us some feedback.

PROFESSOR 1: Come on.

PROFESSOR 2: OK, come on, come on, come on, come on. Who's brave? Yeah, yeah, yeah yeah. You got halfway of your seat. Yeah.

PROFESSOR 1: Yeah, come on.

PROFESSOR 2: I'm sure you have an opinion. Who here does not have an opinion?

PROFESSOR 1: Sure. Yeah.

PROFESSOR 2: Yeah, yeah, good. And then you can line up right-- there you go. OK? There you go. And then others can come up behind as they're motivated.

AUDIENCE: Good afternoon. First of all, I'd like to thank the professors and the TAs for this opportunity. This is absolutely wonderful. And all the students I worked with. You were all great.

I just wanted to say when you were talking about what the most valuable thing in the class people thought was, I don't even know how to pick. Of the projects that we worked on, or that they worked on, I looked at every one and just thought, what of the content of this project have I used just in the past year? And the amount of stuff that I've made significant use of that you guys covered has been at least 75%, maybe even more, of the material of the class.

And that means that if you look over multiple engineers, you probably pretty quickly get coverage of 100% of it. All these projects, profiling, parallelism, algorithms, bit hacks, these are all things that I use on. if not a daily basis, then a weekly basis.

PROFESSOR 2: OK. Good. Thanks. Thank you.

[APPLAUSE]

AUDIENCE: So for me, like, I really had no experience with C or C++ or really distributed version control before this class. So a lot of my time I think was spent kind of learning the basics of that. I spent a lot of time initially learning the basics like pointer arithmetic and that kind of stuff. And then a lot of time learning the profiling tools. So I wish, I really do wish more time was spent kind of-- well, your recitation, I think, would be a good way to really get that out of the way. Because the algorithmic stuff was more easy for me, which I thought was kind of ridiculous. You know?

[LAUGHTER]

AUDIENCE: The fact that these tools are what's holding back, that I have to spend a lot of time learning.

PROFESSOR 1: This is actually something very interesting people have observed. So if you do physics problems or math problems, if you run into an issue, after a certain amount of time you will just go ask somebody. But in programming, people seem to go on and on and on trying to debug this simple thing. And part of that, part of what we try to get is also not just all the cool algorithms and stuff like that. To give you a process. Because there are times when you start saying this program is slow, what you do? And I think that's a missing part sometimes about the process of debugging. If the program doesn't work, what do you do? Where do you start? I think that's why we are trying to do a little bit of, basically, these life skills.

AUDIENCE: It was great to learn but it was more stressful to learn. I'm not as willing to ask as a lot of people, so it's hard for me to--

PROFESSOR 1: So one thing we're trying to do is probably--

PROFESSOR 2: That's great feedback.

PROFESSOR 2: We will try to formalize next year the seventh floor lounge, because a lot of people who came there benefited. But have it a little bit more. And also get a little bit more life skills type things early on. Like if you find a problem, what will happen?

AUDIENCE: It could be a lot longer than that.

PROFESSOR 1: Yeah.

PROFESSOR 2: Yes, so that's our goal, is to make it so that people come up to speed more quickly. But in some sense, another goal but I have here, next year, is that the C students will stay in the class. OK? So what happened is most of the C students dropped out. So in case you want to know, most people are going to get A's and B's. OK? So most of the C students dropped out. And I would like to have people complete the course, even the C students, feeling that it's useful to complete the course.

AUDIENCE: So I think again, as the person before me, I learned a lot. I hadn't had almost any experience in C or C++ before the class. I had some experience with git. But I think that the class has a lot to offer. There's a lot of content covered. And I wish that, for example, there are things that I wish that I could have learned more about, how to implement in a formal way. Things like SSE instructions.

So I feel that like there was a clear cut off of people who had a lot of experience and they could be like, OK, I know how to do assembly, I know how to do other things, and if I want to optimize this function I can, whatever. Write it down. And then there's these other people that don't have that much experience that in a class it mentions this is how you use SSE or compiler optimizations. It's kind of mentioned but not really more in depth.

This is a case of how do you actually do this in real life? I mean, I think maybe the examples given in class were useful but very superfluous. Like you don't really know. I know that I was looking through other projects that were similar to ours for the final. And I think it's a class that you taught in IAP. There was a team of people that did a ray tracer in the multi-programming. And they used SSE and CMD everywhere. They packaged everything. But that's something that I said, it'll be really awesome to do, but I have no idea how to start doing it. And I think that there's still a lot of room to actually apply the concepts that are trying to be taught.

PROFESSOR 2: I think it's our intention actually to increase that part of the course, the data parallel part of the course. Which I think we'll be able to do. One of our concerns about adding material has been, OK, that means you guys have to learn it too. But I think that if we're able to reduce the workload of the learning of the tools by having recitations, we then can look at adding other material, and expect that the load on students won't go up.

AUDIENCE: Right. I mean even what I want to say. It's not even adding material, what I think. I feel that I was going to class, learning all of these cool things. And then I was looking at my project and it's like well, these are the things that I know how to do. And I know that they mentioned all these other things in class. I don't know how to make them in my project.

PROFESSOR 2: Great comment.

PROFESSOR 1: So I think one thing both good and bad about this class is what happened in-- I think it applied to all of you. When you try to teach computer science, you get people who are for the first time they are writing any piece of code more than 20 lines. And you get people who are checking in code for Apache. And you have that entire range of people in the class. So how do you teach everybody? And a lot of times when you give a fixed project, either somebody's completely struggling to finish that, or they are like in a day is done. They are like, so what's this? What's new?

The thing about performance is, asked lot of people, figured out there's no end to it. And you can keep doing it, you can do more, you can do more deeper. So the nice thing about-- one thing about this class is that there's something for everybody in that. So if you are struggling still, if you're learning, you can still get to someplace, learn a lot. But if you have done that when you are in the fourth grade, you can still go and do something more, push that.

So we try to keep that balance in there. So we don't want to bore anybody. And also we don't want a lot of people dropping out of the class either.

AUDIENCE: So continuing on with what other people said, I also ran into the same problems with that. Just the syntax, just the tools. So I guess I'll start off thanking John and Reed, they're really fast and good at what they're doing. And I feel like I've asked a lot of dumb questions, and they're good at answering those.

[APPLAUSE]

PROFESSOR 2: So John and Reed and Josh all took the class last year. OK? And it's amazing, if you ask John how he did on the final project--

[LAUGHTER]

PROFESSOR 2: He didn't do so well. OK? But it's amazing. You'll discover, it's amazing what you guys are going to learn next semester after being out of the class. Because you're on a path to learning this stuff for yourself now. OK? And you'll find that in all kinds of things you do that all these skills, you can keep building them. We will definitely be looking for TAs next fall. And so people who are interested should-- who are going to be MENGs should-- and even if you're undergraduate, you're still allowed to work hourly for the class. In fact, that's what John is doing. So if you're interested, please let us know. Please let us know.

PROFESSOR 1: So you can have this nice experience as a TA.

AUDIENCE: Yeah, so, and then I want to say something that I really love about this course. I feel like it's such an integrated experience. It's, maybe except project one, everything after that is, you get a piece of code, as you would normally, and then everything is within your bounds of-- you can do anything that you want. So I really like how it's not limited to like, oh, we're on this topic so we're going to write a little algorithm that implements this particular algorithm, and then move on to next unit, and then nothing ever comes together. So that's something I really love about this course.

And the next thing I want to say is the beta grading. For project three, I was working with Angela, I feel like from beta to final, because I don't have much experience with C, so it takes me so long to just get the syntax right, to get my own going. So for beta, we didn't have anything working. It doesn't even build. But with two or three more hours, we got it working. It's not super fast, but we kept working on it.

And by the final we have something I felt pretty proud. But the beta grading doesn't take that into account. Like if you are not very experienced with C, it takes you a long time, longer than other people have experienced to get something going in the first place. But beta kind of penalizes you for just not having enough time to do things you want.

PROFESSOR 1: I think it's a double-edged sword. Because if you don't do beta, then you have very harsh. You have one time code, it's in and out. If you just really don't do, give that much grading for beta and make it optional and stuff like that, people just drop beta. They don't just do anything in it. So we try to do something, give everybody a true chance and get everybody a real motivation to actually work for both sides. Because for a lot of you guys, beta defined and you learned a lot. Because you got something in beta.

And for most of the classes I have, we had things called checkpoints. For example, compiler class, you had to do a checkpoint and checkpoint will be used more like subjectivity to grade. We don't say how many grade percent checkpoint. But if checkpoint has nothing, then we'll be very harsh grading later. We always provide that. But still, people don't take checkpoint that seriously. But here we have to try to make something to make people get beta seriously, do something there, and then give them the opportunity to improve. I think there's a balance in that somewhere.

PROFESSOR 2: And I wouldn't worry about your grade.

[LAUGHTER]

AUDIENCE: Maybe like one idea I was thinking is maybe there's some component of improvement from beta to final. Like the proportional improvement that you made from beta to final.

PROFESSOR 2: Oh, that's a good idea, actually. That's a good idea. Yeah, to say, if you made a-- Although you want to be careful about people sandbagging.

[LAUGHTER]

PROFESSOR 1: Exactly.

AUDIENCE: And then one last small thing--

PROFESSOR 2: I want to make sure we get to everybody.

AUDIENCE: OK, sorry.

PROFESSOR 2: So if you want to go around to the end of the line.

AUDIENCE: That's actually mostly all of it.

PROFESSOR 2: OK. Thank you.

AUDIENCE: So I've said some of this on my evaluation, which I already turned in. But I'll repeat it for everyone else.

[APPLAUSE]

AUDIENCE: A couple things. One relating to the grading which you mentioned. Honestly, I found it incredibly stressful knowing that it was competitive grading. Because we had no idea where the benchmark was. So it's like, well we've improved it. But did we improve it enough? Did everyone else not improve it any more than we did? And it was just like having it be a complete unknown was pretty stressful. So I don't know if there's some way you could do like a-- like maybe the TAs got some speed up, and that's the benchmark to start with or something.

PROFESSOR 1: I guess what we did this time is basically we gave the best grade as a benchmark for the final. And if you go above that, it's just bragging rights.

AUDIENCE: I mean, when you're working on the beta, you have no idea what's going on. But I did like that. Once you know where the benchmark is, then it's nice to know what you're aiming for.

PROFESSOR 1: So the thing about that is, the flip side of that is, we don't want somebody that a lot of courses they say OK, I'm done. When you go there something, Oh yeah, I met the thing, OK, I'm getting a good grade, OK, I move on. I mean, we want people kind of to keep pushing in time. Because sometimes the thing is-- If I have a minute, I'll tell the story.

There's this guy named Tony Danzig who basically invented the simplex method. His thing was, he was always late to class. He went to class one day and there was-- normally the professor writes the problems on the board. So he copied the problems down and went home. And they were really hard problems. And so he was like, damn. I don't know how to solve this. And he worked very hard and solved it. And wrote a note to the professor saying I'm sorry this is late, and put it on his desk.

And a couple of hours later he gets this excited professor running around trying to find him. Because what the professor had done is written two unsolved problems on the board. He came late to the class, he didn't know that, he had two problems, he copied them down. He worked very hard and solved it.

[LAUGHTER]

PROFESSOR 1: And so the professor said divide this up, you get your pay at the end. Then there's lots of it. So this is something like that. If you give the thing, if you know where we are going, for some people it will be a hard climb. But other people, they will get there faster, they'll say OK, I'm done. And what we want to do is give them the ability to kind of really push to their limits, not some artificial limit. But not to penalize too much for that. We try to balance this out. So I want everybody to go to their best limits.

AUDIENCE: I had one other thing to say. Have you considered or would it be possible to make the class more than a 12 credit course?

[LAUGHTER]

AUDIENCE: No, seriously. Since there is so much material and we keep talking about wouldn't it be nice to add this and add that. But I don't know about you, but I spend a lot of time in this course already. And I mean, I think it's the kind of class where you could pack 16 or 18 credits worth of material into it.

PROFESSOR 2: More realistic would be to have a follow-on.

PROFESSOR 1: Try to make it as two classes.

PROFESSOR 2: To make a two-semester sequence. But I think Saman and I still feel that this, it's like, OK, that's a new development effort. That this class still-- I think we feel that there's more than incremental improvement we can do to make the class teachable. For example, by introducing recitations. And so I think once-- maybe after next semester, next fall we can we can-- But thank you, because I think that that would be good.

And the whole idea of the sections is to try to reduce your load. There used to be a thing that was on all the evaluations that they no longer do. And this is actually before I got to MIT. They had a measure called cums per tooling hour. OK? So a cum is how many units of knowledge you're using in whatever measure, and tooling hours, like how hard you work. And they got rid of it because they said it was not quantitative. After all, what's a cum? All right.

Well, regardless of the fact they got rid of that, because I think that it actually was a very useful measure, even if people are subjective about it. That's our goal in the course. It's not to make people spend a lot of time and then have learned epsilon. It's to have them, for the time they put in, learn as much as possible. And so I think that we recognize it's a lot more work right now that you folks are going through. We hope that it's rewarding work. But we hope we can do even better in terms of giving people the opportunity to learn these things.

PROFESSOR 1: So we are thinking about cutting things. We all are thinking about what to eliminate even if we add something. It's very hard because we think a lot things are useful--important. But something has to give. So that's the hard decision.

PROFESSOR 2: Yeah, we're thinking of cutting caching and parallelism.

[LAUGHTER]

PROFESSOR 1: OK.

PROFESSOR 2: OK. We have to hurry because we gotta be out of the room by 4.

AUDIENCE: I just wanted to say that I agree. That I didn't really like the way the course was graded. But I did like the material in the course a lot. So that was good. But for one thing, the thing about the competitive grading was that it was way more stressful than necessary. My luck was good because we had the kind of benchmark thing. It was a reasonable high performance to attain.

And also I didn't like the way some of the stuff was weighted. For example, it didn't really correspond to how much time I spent on it. The pset was worth like 3% or 4% of the grade and it took like hours, I think. Many, many hours. That's all.

PROFESSOR 1: OK, thank you.

PROFESSOR 2: OK.

AUDIENCE: There is one thing that's pretty general about course six, I guess, is that they're really lenient on prereqs. Since I've only been at MIT for like three months.

[LAUGHTER]

AUDIENCE: So yes, I'm a freshman.

PROFESSOR 1: One freshman in the class. There you go.

AUDIENCE: I'm really thankful that Professor Leiserson and Professor Amarasinghe let me into the course without any prereqs. And I like the breadth of the topics. But I think with office hours we could have gotten a bit deeper into them. And one suggestion I'd like to make is that get the nightly builds working. It was there for one night for project one, and that was it.

[LAUGHTER]

AUDIENCE: So I decided this is my last--

PROFESSOR 2: Can you talk in the mic?

AUDIENCE: So this is my last term at MIT. And I decided to take things easy by not taking that many courses and TAing this course.

[LAUGHTER]

AUDIENCE: So yeah. It would be nice to have 48 unit TA shifts too.

PROFESSOR 2: Thank you.

AUDIENCE: I just want to say that this was like-- should I take it?

PROFESSOR 1: Sure.

AUDIENCE: OK. That this was like an amazing class. I think everyone can agree that we learned so much that is not found in textbooks. And what I thought was really weird is that every little random thing I've ever read on the internet related to coding, I guess, has come up in this class in like, useful, come up like useful in this class. Like every little bit of outside knowledge that you take yourself has appeared useful in this class later on. And I thought was really cool.

And related to the hardness that people are talking about. I think everyone came into the class bracing for the absolute worst, because we've heard about you guys.

[LAUGHTER]

AUDIENCE: But, yeah. I was actually really pleasantly surprised that there were helpful TAs waiting in the G7 lounge for you, even before office hours start. And so I was actually really pleasantly surprised by-- just like everyone, the class. I feel like everyone who has survived until this point has some sort of magic skill. That you really learn a lot by working with them on the group projects. And so I was just really inspired. And I thought it was a great class.

PROFESSOR 1: Thank you.

PROFESSOR 2: Thank you.

[APPLAUSE]

AUDIENCE: Hi. I love the class. I appreciate you Professor Leiserson, and everything that everybody else before me said. I just wanted to mention two things I don't think were mentioned. Beta programming, to be honest, was amazing because except the final project, everything I did was beta programming in the projects. And the amount of bugs that we avoided because of that is ridiculous. Especially when we're doing pointers, and if it doesn't work, make it a pointer, and all those philosophies that we came up with. That's the first thing.

Second thing is, and I've already mentioned this. A book or readings for this class or links. Because as amazing as the lecture slides are, a lot of work done there, so thanks a lot. They're hard to connect together, especially when you're trying to figure out what our cache-oblivious algorithm is trying to do with those. So that's pretty much it. Are you telling me the stack overflow is as much as this course?

Well, thanks a lot again for the TAs. Most of my learning came from them, to be honest.

[LAUGHTER]

[APPLAUSE]

AUDIENCE: Hands on learning. Hands on. Thank you.

[LAUGHTER]

GUEST SPEAKER: Hi. my name is Harold Pokob, I'm the head of engineering at Akamai. So I wanted to use the opportunity to thank all of you for taking the class. Charles was my adviser when I was at MIT and he told me about his plans for the class. I thought it was a great idea. Getting a lot more hands-on experience going. Back when I spent time at MIT, it was a lot more theory, and Scheme was one of the main programming languages. So in that context I really like this type of class. And I'm glad MIT is investing in that and supported from Akamai. So I'm honored that we could sponsor it. I'm very happy you're here. There is, I think, refreshments downstairs in a few minutes.

PROFESSOR 2: Yeah. I'd also like to ask Harold to say what the topic of his master's thesis was.

[LAUGHTER]

GUEST SPEAKER: My master's thesis was cache oblivious algorithms.

[LAUGHTER]

PROFESSOR 2: So you have Harold to thank for one of the papers that you read during the term.

GUEST SPEAKER: It's nice that something that I did in academia is still used. Not all now practical business stuff that I do these days. Great to have you here, and I hope to see some of you downstairs. And I've got to actually go into a meeting at 4:00 for half an hour, but I'll be back down afterwards. Thank you.

PROFESSOR 1: Thank you very much for sponsoring this event and getting all the food.

[APPLAUSE]

PROFESSOR 1: OK. Now we have food.

AUDIENCE: We also have quizzes here.

[LAUGHTER]