WARD: Well, good morning. My name is Steve Ward, and I have the intimidating honor of chairing a panel of distinguished Turing Award winners. As you know, the Turing Award is the most prestigious award available in computer science. And our panelists include Professor Andrew Yao, who has been on many faculties, including MIT's, and is currently from Tsinghua in Beijing.

MIT's Ron Rivest, and MIT's Barbara Liskov, and Butler Lampson, who hails from both MIT and Microsoft, and Professor Fernando Corbato of MIT. I thought we would get started by asking each panelist, in turn, to say a few retrospective words about the work that earned them the Turing Award, perhaps shedding a little light on their motivation and what their retrospective view is of the impact of their work. And I thought we would do this in kind of chronological order.

So I'll start with Corby, who was cited for pioneering work, organizing the concepts and leading the development of the general purpose, large scale timesharing and resource-sharing computer systems, CTSS and Multics. Corby?

CORBATO: Thank you. Well I'm reminded that, first of all, that it was 50 years ago this spring that MIT had its 100th anniversary. And right here in Kresge, there was a series of symposiums, one of which was chaired by Martin Greenberger of the Sloan School, who had a symposium called Management and the Computers of the Future.

That was published, eventually, in a book by MIT Press, and it got relabeled Computers and the World of the Future. It's still in print, actually, thanks to demand printing. And for $40, you can get it.

Most notably in that particular session on computing, there was a lecture by John McCarthy, who subsequently went to Stanford, and got his Turing Award also. But his seminal work began here at MIT. But he, in particular, in his talk, in that symposia, advocated timesharing, and made a good case for it. And the notion of a computer utility.

Well the other thing is that computing 50 years ago, was only a little over 10 years old, modern computing, that is, programming. And the thing that was staggering at the time was how frozen the attitude of the industrialists were, and the computing manufacturers. They really didn't understand why they needed to change anything, and timesharing was against the grain.

And so CTSS was started out. It was almost kind of a hack. It was going to be a demo. It gradually, as it succeeded, grew and grew. And it was quite successful.

One of the other things I would notice is that timesharing was the phrase, but the real goal was man-machinbe interaction, or as Licklider said, the man-machine symbiosis. There were two key people in that evolution. One was Licklider, who was an inspiring advocate. And Bob Fano, my friend, who was also very wise and the leader who had the leadership to start Project Mac, from which CSAIL evolved directly. I think there's a lecture. Fano gave a presentation in anticipation of this celebration, which is on the web, I believe.

And then finally, a last remark would be that CTSS and Multics, which succeeded it, were precursors to the contemporary systems and networks. And all of the interfaces and ideas that have shown up now and set the stage for personal computing and the altos-- which Butler, I'm sure, will talk about-- in the PC world that we know today, got started in timesharing as people began to learn to interact in a real-time basis.

It also led to things like Unix, and the CC programming language, and ultimately today is being rediscovered in cloud computing.

WARD: Well, thank you, Corby. And if Corby was the consummate system builder of the 1960s, I think Butler Lampson certainly deserves that honor for the 70s. He was cited in 1992, when he earned his Turing Award, for contributions to the development of distributed personal computing environments, and the technology for their implementation. Workstations, networks, operating systems, programming systems, displays, security, and document publishing. It's hard to know what got left off of that list. Butler?

BUTLER: I can tell you exactly what got left off. Spreadsheets and the web got left off. When Xerox started the computing part of the Xerox Paolo Alto Research Center 1969, their short-term motivation was that they had just made a foolish mistake and bought a computer company. Their longer-term motivation was that they were able to see that the light lens copying business, over which they had a monopoly at the time, was going to be very nice for a decade or two. But it was not going to last forever. And the senior management of Xerox thought that they ought to prepare for the future.

So they started this lab, and they told us to invent the office of the future. Of course, they had no idea what that meant. And neither did we. But it turned out to be a very fruitful instruction to give us, because based on that, we were able to develop pretty much all of the things that you're now accustomed to in the world of personal computing, networking, and local file systems, and drawing programs, and document editing.

The first internet was actually built at Xerox PARC. For some mysterious reason, Xerox decided that we shouldn't publish anything about it. So the people who did the work never got much credit, but they did go to the seminar at Stanford where [? Vint Surf ?] was developing internet protocols. And they weren't allowed to talk about the internal network, but they were allowed to ask questions. And they asked a lot of questions, and finally [? Vint ?] looked at one of them and he said, you've done this before, haven't you?

So we did all that stuff. We didn't do spreadsheets. That would have been technically possible on the equipment that we had. But we had no personal use for spreadsheets, since although their use has expanded a lot more recently, originally they were meant for accountants, and we didn't do accounting. And we didn't get along very well with the people at PARC who did do accounting. So that's why we didn't do spreadsheets.

And the reason we didn't do the web was that we didn't have a big enough sandbox to play in. My theory about the reason that the web came along when it did is that that was the first time the internet was big enough to make it worthwhile for people to actually post things on it.

Anyway, we did invent all this stuff. And it's pretty much the direct predecessors of all the stuff that people use today, even the first quote, laptop, unquote, was built at PARC in the late 70s.

WARD: OK, thank you. Andy, you were awarded the Turing Award in the year 2000 in recognition of fundamental contributions to the theory of computation, including the complexity-based theory of pseudo-random number generation, cryptography, and communication complexity. Can you give us your retrospective view of that work, and its impact, and your motivation?

YAO: I think my Turing Award was mainly due to the proposal of two concepts around 1980. These concepts would lead to the creation of two subfields in theoretic computer science, which over the years have shown a lot of depth and applications.

And the first of these is communication complexity, which captures the amount of communications needed when several computers want to perform a joint computation based on their private inputs. And the second concept that I proposed was about the perfect pseudo-random number generator.

Pseudo-random number generation had been a very important subject in simulations and in other fields. And the standard statistical theory about it is that whether a pseudo-random number generator is good or not depends on the application. And typically, you have some sort of mathematical or statistical test for seeing whether it's good or not.

Now with the advent of cryptography, we have to be more stringent, and we have to think about is there a way so that we can define a good definition of pseudo-random number generator so that it can satisfy the very rigorous requirements? And so I proposed a definition, which is I think it's pretty outrageous. I was asking the question, is it possible to have a pseudo-random number generator that can pass all the statistical tests that people can think of?

And it actually turned out to be the case with the help of computational complexity, and myself, and many of my colleagues, together we have created to make it work. And now in retrospect, I think it probably should not have been a surprise that concepts like these would lead to ramifications. And the reason is that around 1980, there were two seminal intellectual trends that started to happen in computer science.

One is the change from single machine to network setting, and I think [? it's ?] pioneered by people like Butler and others. So the cost consideration would no longer be just the computation time for one single machine, but actually you have to take the communication costs into account. And the second seminal intellectual trend that was started at the time is that at that time, the proposal of the public key cryptography led far-sighted people to think about, in the future, maybe we will have a big network and we can do electronic commerce, and cryptography will be very useful and important.

And now when you think about cryptography, the traditional theory was the Shannon theory of cryptography. But Shannon's theory was very rigorous. However, it's perhaps too rigorous. If you insist on that security, then you practically cannot do any electronic commerce.

And so I think at that time, near the end of the 1970s and the beginning of the 1980s, many of us were struggling to try to break away from the Shannon cage. And instead, we would embrace the computational complexity to become our Savior so that we can construct frameworks so that cryptography will be rigorous and will be meaningful.

And so I think that given those two trends, I think that it should have come as no surprise that those of us who were lucky enough in those early days who started to think about such issues would come up with concepts that would become very important.

And perhaps I should explain why I am here. I think all my distinguished friends here are MIT faculty members. I think that firstly, I was a faculty at MIT during 1975 to 1976. And secondly, I think my work has been influenced by a lot of the MIT people who are here today. I think, for example, in the communication complexity that Harold [? Abelson ?] and the pseudo-number generation by Ron Rivest, [? Shafi ?] [? Goldwasser ?] and Sylvia [? McCauley. ?]

And I think there's an interesting story that, I think, Ron if you forgive me, that I will just repeat it. I think Ron always joked with me that I missed the boat, because I left MIT at 1976, otherwise I would perhaps be able to get a Turing Award together with him. But fortunately, I went to Stanford, and then later on had the opportunity to mix with the Berkeley crowds. And so I guess I got into cryptography on the west coast. And I guess everything is probably preordained, after all.

WARD: Thank you, Andy. Ron, you were awarded the Turing Award in 2002 for ingenious contribution for making public key cryptography useful in practice. Your retrospective view?

RIVEST: Yeah, sure. So Andy's talked about some of the themes that were around in those days. So our work was inspired by the [? Diffy ?] [? Helman ?] paper talking about public key cryptography. And it asked the question whether public key cryptography was possible. And with Len Edelman and Adi Shamir, who were also members of not CSAIL, but LCS at the time, although members of the math department, and that shows the strength of the interdepartmental character of the enterprise of computer science.

We came up with a proposal known as the RSA scheme. Now I guess if you'd stayed around, it might have been RSAY or--

YAO: Or YRSA.

RIVEST: YRSA. [INAUDIBLE] letters. YRSA. Whatever, yeah. So anyway, we came up with a proposal, which has so far stood the test of time, based on the difficulty of factoring the product of two large prime numbers and doing some modular exponentiation for encryption and for decryption. We still don't know, in the end, whether it's secure or not. We have many open problems in the field of computer science and theory. And showing that a problem is hard remains one of the most challenging problems we have, and we don't know whether specific problems like factoring are hard, or particular cryptosystems are hard.

But the cryptography has thrived. And one of the impacts that our work had, I think, is to really help establish the character of the questions that people ask. And the kind of questions you can ask and answer is can you design cryptosystems so that the difficulty of breaking that cryptosystem is closely-related, or identical to, some hard problem that you believe to be hard. You make explicit assumptions that factoring is hard or discrete logarithm is hard, or doing discrete log on elliptic curves is hard, or something like that.

And the field has really blossomed in a very nice way, in spite of our inability to show that these particular problems are hard, because we can make very crisp assumptions saying, if you assume that these problems are hard, then you can achieve these cryptographic objectives. And so the field has really blossomed in a wonderful way, and achieved a lot of paradoxical things.

I think the public key work, itself, demonstrates the paradoxical character that cryptography can have, that you can ask something which seems like you shouldn't be able to do it, and yet you can in the end with a clever use of the tools at hand. And that's a theme that appears over and over again, is that by making the right assumptions, by being careful about your reductions, and asking what appear to be paradoxical questions, a paradoxical set of requirements, you can actually achieve some of these things. I'll stop there.

WARD: Ok, thank you, Ron. And Barbara, your 2008 Turing Award cited contributions to practical and theoretical foundations for programming language and system design, especially to data abstraction, fault tolerance, and distributed computing.

LISKOV: OK, well Corby started by mentioning John McCarthy, so I wanted to say that John McCarthy was my thesis advisor at Stanford. And I was working in AI at the time, but as soon as I got my PhD, I switched into computer systems. I went to work at the Miter Corporation.

And while I was at Miter, I started to work on program methodology, because there was a huge concern at the time-- and, of course, there's still a concern-- about what they then called the software crisis, by which they meant we had no idea how to build software that actually did what it was supposed to do. And I found this problem very intriguing. And so I was thinking about it, and I moved to MIT in 1972, and this was the problem that was consuming me. What would be the right way to organize software so that we could actually control the complexity and reason about the resulting programs?

And I had in mind a particular way of organizing programs, which I referred to as a multi-operation module. So the idea was that I had a big black box, and inside, there was complicated data structures, and very complex code. But it was all hidden inside the box, and the box provided an interface that consisted of a number of operations. And outside the box, you could call these operations, but that was the only way you could interact with the box.

And this seemed to me like a really good way to organize programs. And there were other people that had similar ideas, like Dave [? Parness ?] at the same time. But I felt that it was very hard to communicate this idea to normal programmers.

And so in the fall of 1972, shortly after I got to MIT, I had this wonderful aha moment. And very few people are lucky enough to have a moment like this. And what happened to me was, I all of a sudden saw that I could relate this idea of a very abstract multi-operation module to data types. And I felt that making that connection was going to be really beneficial, because programmers understood data types. They program with data types all the time.

And the connection was that the module was a data object, the operations were the operations of the object that you used to manipulate it, and people already use things like arrays and they understood that you could add something to an array, remove something from the array. And they also understood that they thought about these things abstractly, and did not have to actually understand how things were implemented underneath.

And so I had this idea of abstract data types. And then, of course, you know you're lucky enough to have an idea like that. But, of course, it doesn't stop there. I spent several years trying to figure out with this idea really meant. How did you define it formally? What was an abstract data type, conceptually? How did you specify them? How did you reason about their correctness? How did you design programs using the idea? And so on.

Around 1980, I switched into distributed systems, so that was the other part of the Award. And so that was just an example of what I was doing in my career. I was opportunistically switching from one field to another as I saw really important problems. And when working in distributed systems where I work today, I got very interested in replication, and so forth.

So I've had a lovely career. And I was lucky enough to have some good ideas. And that's where I ended up.

WARD: Well, thanks, Barbara I have here a handful of questions that have been submitted by attendees. And I'm unlikely to be able to get to them all, but let me start with several ones that I think should be easy that deal with the institution of the Turing Award, itself.

The first one is in 25 words or less, how did the Turing Award change your life? Anybody like to respond to that? OK, well if you decline responding to that, let me get to one that's even shorter and considerably more blunt.

Are Turing Awards really a good idea? I'm guessing this was submitted by somebody who doesn't have a Turing Award. If I had to guess, I would be able to guess a one-bit answer that each of the awardees would give.

BUTLER: Is this meeting a good idea? I think they're very closely- questions.

WARD: Perhaps. But I make the assumption that none of you did the seminal work that you did specifically in order to win a Turing Award. OK? So if it didn't have that effect on your behavior, did it, in fact, have any effect on your behavior? As a social institution, is it a positive thing? Anybody want to comment on that?

LISKOV: Well, it's not for individuals. You don't do research because you're thinking about the Turing Award. You do research because you find things that you are fascinated with that are interesting, useful to work on. It's probably good for the field, because it gives us a history. It recognizes contributions. It gives people a way of thinking about the history of the field.

Of course, it's not perfect, because the people who get the award, they have to be nominated, the nomination has to be phrased properly, the committee has to be receptive. And so clearly, there are going to be people who didn't get the award who deserved it. And there might be people that got the award who didn't deserve it. You know, who can say?

But I think if you think about it sort of on average, it's a useful thing for the field.

WARD: Is it fair to have the model like tenure? People who have it think it's a good idea.

Well, let me move on to something that's closely-related.

RIVEST: Let me toss out an answer to that if I may, too. I do have an opinion on it. I think the Turing Award is great, and it adds some value to the field. But I think more important to the field are awards going to younger people before their promotion, and tenure, and so on, too. I think that that's a critical thing to recognize accomplishment early in people's career, and the Turing Award tends not to do that.

WARD: OK. A related question is can you think of work that has already been done that deserves a Turing Award, and hasn't yet received one? For example, would you like to make any predictions as to the 2012 Turing Award winner? You, after all, are pundits, and you need to exercise your crystal ball a little bit.

CORBATO: I'll step into that one. I'm not sure I know exactly who to credit, but I think there probably is not a person in this room that doesn't recognize the profundity of the contributions of Google. It's become a verb. People have thrown away their encyclopedias. And it's a wonderful contribution to our well-being, and so forth.

It had a predecessor, which isn't well-known, Altavista. Butler, maybe, can talk about that. Google has succeeded for reasons I don't fully understand myself, but it's miraculously fast.

BUTLER: It's fast because they have a lot of money. And they're clever too, but it takes a lot of money. And the reason they have a lot of money is that they figured out what very few people did, which is that you could make a lot of money out of targeted advertising.

WARD: Well, aren't there similar institutions? I'm thinking of things like Wikipedia that doesn't have a lot of money that's also had a significant impact.

CORBATO: Yes, I'd put that in the same ballpark.

WARD: So money isn't essential? OK.

BUTLER: Whether money is essential depends on what it is you're trying to accomplish.

WARD: Perhaps.

BUTLER: For going to the moon, you need a lot of money. To index the internet, you need a lot of money.

WARD: OK. Let me move on to another question, which may be viewed as superficial or profound, depending on your view. But it should be easy to answer it. Actually I'll read it in its entirety. It's addressed to each of the panelists. And the question is this-- do you have a Facebook account-- I haven't finished-- a Twitter account, and why not?

So, Barbara, fess up. Do you have a Facebook account?

LISKOV: Absolutely not. Neither Facebook, nor Twitter.

BUTLER: Do you have an email account?

LISKOV: Of course I have an email account. I'm not particularly interested in that kind of exposure. I'm not really interested in following people, or having them follow me. But I'm a person of my generation. And if I were 40 years younger, I would probably have a completely different answer, because I do think it's generational. We grew up in a certain way. We're used to a certain way of living. And you sort of settle into a rut. Yeah.

I do think, though, that there's something quite dangerous going on in the internet, and Facebook is a good example of that, because of the way people expose themselves, not understanding how they might regret that later. And I'm keenly aware of the dangers of the internet. And thank God for Ron and all the crypto so I can at least do online banking.

But I'm very careful even with email and what I will send, because-- and you'd think politicians would learn this after a while-- because everything you do like this is exposable, and I just don't care to be exposed.

RIVEST: I do have a Facebook account. I do have a Twitter account. I make sure to log into Facebook at least once a quarter. Twitter at least once a decade.

WARD: So my follow-on question for you, Ron, is what are you going to do about the 600 friend requests that you get as a result of that announcement?

RIVEST: No promises.

WARD: OK. It's sometimes argued that all branches of computer science have enjoyed success because of the remarkable, exponential growth, the Moore's Law growth for over five decades. And so an interesting question is would your work be as celebrated as it is now? Would you even have done it if it hadn't been for this Moore's Law growth, this exponential growth, and the impact of computers themselves? And there is kind of a rising tide, floats all boats argument that maybe good research has turned into great research because of the economic impact of all of this growth?

BUTLER: Certainly true for most practical systems work. We couldn't have done what we did at a PARC five years earlier. The hardware would have been impossible to build.

WARD: Were you excited about building that hardware in anticipation that someday it would become cheap enough that people could put it in their pockets, and have it at home?

BUTLER: Absolutely. I wrote a guest editorial on this subject in 1972. It was crystal clear that that would happen. But I think it's fair to say, I suppose from the point of view of the welfare of the Xerox Corporation, that was quite essential. We would probably have tried to do the work anyway, even if we couldn't have foreseen that future, but we certainly did foresee it.

LISKOV: Yeah, backing up what Butler said, during the 70s and 80s, everybody working any central area of computer science was keenly aware of the fact that computers were getting faster and faster. And so you could envision doing things today that weren't going to be practical today, but you were laying the framework for something that would be practical a little bit down the line.

YAO: Yeah I think even for theorists, I think that it's very important to do theory problems that are related to technology. Somehow the advances of technology bring out a lot of really interesting problems. So I don't know whether it's a coincidence or there is a correlation that somehow things that have very heavy technological content also have very rich theoretical content.

And I think that I was pretty lucky in getting into this field. And I have been quite interested in combinatorial problems like [INAUDIBLE] problems. And all those problems were very fascinating to me at the time when I entered computer science. And I couldn't really decide whether to working combinatorial problems or doing something else, like computer science.

And at some point, it suddenly hit me that there were so many interesting mathematical problems in the world that how do you decide which problems to work on? And computer science, being a very rapidly-advancing field, kind of breaks the symmetry. Somehow it singles out problems that is different from the typical mathematical problems. And I thought that was a good choice. And I still don't understand whether there's a correlation, but somehow it seems so.

WARD: Well, to put it more indelicately, was there a bit of technological opportunism in your career choice, then?

YAO: Not really. I think that, actually, I first got into theoretical computer science from [? Don Kanut's ?] books. And so I think that when I first read the manuscript and did the exercises and the open problems, I didn't really think of the opportunism in technology. But somehow later on, when I thought about deciding whether to pursue computer science or not, then I realized that besides offering interesting problems, computer science also is a new discipline.

I think that, at that time, I understood computer science was very, very new compared with mathematics. But now, after 40 years when I think back, I realize how close to the beginning I was, entering into that field. Because when I entered that field, there were already many big names and people. But I think that it's very important to get into an area early. So I think that perhaps all of us have that advantage here.

CORBATO: I think the evolution of the technology was kind of foreseen by most people as in a misty way, without recognizing everything. Clearly some of the mini computer vendors didn't see the impact of the processor on a chip very well. They're out of business. But from the earliest days of timesharing, we had already discovered social interaction in the form of messages to each other, which later evolved into email. Initially, they were just messages within the system.

And today the social interaction extends to Facebook, and now even Twitter. Yes, I do have a Facebook account, but I'm like Ron. I don't I don't log in very well. I was coaxed into it by my grandchildren. But I, too, share with Barbara the hesitation to expose more than I have today on my website, which is a brief bio, which is comparable to what you can find just by any normal Google search.

So I think some of those reservations are because maybe we've been around long enough to recognize some of the disasters that occur when people make mistakes. But I think most people have that aha moment when they send an email, they regret it. And they realize that they can't get it back. And there's nothing they can do. And it was a revelation to everybody when they first discovered it.

WARD: Well, maybe the next Turing Award winner should be the person who [INAUDIBLE] the unsend key.

RIVEST: Steve, I wanted to respond to your question, too, if I could, because theory is a bit different I think. And the ability to do theory, untethered to the technology of the day is a characteristic of theory, sort of by definition. And we see that today with quantum computing, right? It's not clear at all whether we can build quantum computers. It may be possible.

But there's a wonderful theory evolving. Scott Aaronson and other people are working on this. And it's an example of theory evolving independent of the technological underpinnings. And I think, in response to your question, had the computer technology frozen in the 60s or the 70s, we would have seen much of the same theory evolve, too. Maybe there wouldn't have been as much funding or something. I don't know. But it would have been kinds of questions asked and the kinds of evolutions might have been very much the same.

We call it empty completeness, so it came about about that time, too. Of course, empty completeness is sort of on the pessimistic side of theory. It's [? sort of saying ?] what you can't do, maybe. So you don't need the hardware to do that. But even public key was ahead of its time, in the sense that it really didn't have a utility until the web was invented.

WARD: But if the market hadn't developed for public key cryptography, you think you still would have done it? You would just be an obscure number theorist instead of a Turing Award winner?

RIVEST: There was a conference series started in the early '80s, but the market for cryptography didn't really happen until the 90s. So there was a whole decade of vigorous theoretical exploration before the market happened there. So theoretical work and cryptography is part of that.

BUTLER: There's been a big market for cryptography for hundreds of years.

RIVEST: Well, big in the sense of-- the military market's always been there, of course.

BUTLER: [INAUDIBLE]

RIVEST: Yeah.

WARD: OK. Before leaving the Moore's Law thing, can we look at, potentially, the dark side of it? Is it possible that the Moore's Law growth of technology has made too much function available, too fast, that we haven't actually had time to develop mature engineering disciplines? There's a lot of complaints about the quality of software these days, for example.

BUTLER: It's a law of nature that there's always going to be a lot of complaints about the quality of software. Why is that? Because we always raise our aspirations to the point where we can't quite do it, or we can just barely do it. So that's just-- words fail me.

LISKOV: But I actually think that the much more serious implications are the sociological implications of computing. I mean, we heard yesterday in that talks about the financial markets about the danger of completely computer-controlled trading. And I think if you look across our technology, you see this in many sectors. That we have developed the technology. I mean, I don't think the problem of the software is the big problem. I think it's the control of technology.

But, of course, this is not limited to our technology.

BUTLER: Yeah, it's not as bad as nuclear fission. Just look at it that way when you start wringing your hands.

LISKOV: But nevertheless, it's a serious problem. And it takes time for society to catch up with the implications.

CORBATO: I agree with you, Barbara.

BUTLER: It's not as bad as the [INAUDIBLE] either.

WARD: OK let's exercise your crystal ball a little bit more. Put yourself in the position of a new PhD student, just starting your career. What problem would you choose to work on? What do you think are the important problems that are facing computer science now, and the ones where there is still dramatic progress to be made?

BUTLER: I'll give the same answer I gave last time a similar question was asked on a panel. P=NP. That always silences everybody for a short time.

WARD: Would you actually, as a PhD supervisor with a new PhD student, would you advise him to tackle, as a thesis topic, settling the P=NP question?

BUTLER: No, that would be unwise.

WARD: That would be bad career advice.

BUTLER: It's not quite what was asked.

YAO: Well, I would advise a really brilliant PhD student to work on the factoring of large integers by classical means, which may irk Ron a bit. But I think that that's the right problem.

RIVEST: It's a question we want to know the answer to.

YAO: Yeah.

LISKOV: Yeah.

RIVEST: I think machine learning is a really promising area. It's vibrant right now. There's a lot happening, and I think new students with a lot of mathematical or other skills could make progress there. And I think that's an area which could be recommended.

BUTLER: Well, Ed [? Lasaska ?] already gave my spiel on the subject yesterday. I think it's very clear that all the sort of classical simulation and communication applications of computing will continue to be interesting and important. The really hot area is going to be what I like to call embodiment. And what the NSF calls by the bureaucratic name cyber physical systems. Which to me, means computer systems that have non-trivial interactions with the physical world. This is something that's just getting started, and it's going to be even more interesting and important than the areas that have already been mined.

LISKOV: So I thought Ed's talk yesterday was great. And you gave me hope about systems again. I think that the characterization of systems going up and down the trough of despair and so forth, I think that's really right on. And it's very hard to answer the question for any of us, I think, of what we would do today, because we're not young anymore. And when you're young, you see the world a little differently than when you're not so young.

My thought about today is that I still think there are very interesting issues and systems. But I also think that it's not really embodiment as much as the combination of computer science and something else. Like I do think computational biology is really interesting. And so if I were starting off today, that's the kind of thing I might do.

I think all of us benefited from the fact that we happened to see this new field. And we were lucky enough to get in at the beginning of it. And I think that's a way that you have a really interesting career, because there are just so many open problems that you can work on that people have not solved before. But it's very hard for us today to see what those--

WARD: Well, isn't the flipside of that argument, to put it very cynically, that you guys have taken all the low-hanging fruit? You've built these great systems in the 60s and 70s, and all the new PhDs can do is kind of fine-tune the fundamentals that you've laid down?

BUTLER: Absolutely not.

LISKOV: Somehow they find interesting things.

BUTLER: That's right.

WARD: OK. So another question from an attendee, directed to all, and it reads, isn't it time for parallel programming techniques that mortals can understand?

BUTLER: It is, indeed.

LISKOV: Yes, it is. That's a really important question. And it's not clear to me that the answer is yes, that in the end, we will succeed. And if we can't succeed, then that's going to limit the applicability of the huge multi-core computers. And that's not necessarily bad. It just means that the technology will go in one way or another. God did not say that computers had to have thousands of cores. But I think, as a technical challenge, it's a huge, interesting problem right now.

WARD: Is it a programming language problem, do you think?

LISKOV: I think so, yeah. It's both a programming language problem, and it's a methodology problem, because programming languages follow methodology. So it's a methodology problem. If we can figure out how to organize those programs, that's the first thing we have to do. Then we can invent programming languages that support the methodology. But so far, we don't have too much of a clue about what to do with this.

BUTLER: There's one striking counter-example to that, which is MapReduce.

LISKOV: Yes, but I don't think MapReduce goes all that far in solving the general class of problems.

BUTLER: It goes nowhere in solving the general class of problems. That's certainly true. But there is a large class of problems where we actually do have a reasonable answer.

WARD: Well, should the question be, actually, limited to parallel programming? I mean, what about just everyday programming? One could ask the skeptical question that, despite all that you've taught us about programming languages, Barbara, JavaScript has emerged as the most popular programming language in the world. And that's hardly an exemplar of your lessons. Can that be interpreted as evidence that programming languages don't really matter that much?

BUTLER: The correct way to look at JavaScript, is it's a machine language.

WARD: Well, it's a machine language that a lot of lines of code are written in every year.

BUTLER: Only for relatively small programs. People want to write--

WARD: That's changing.

BUTLER: People want to write big programs. I don't think so. People want to write big programs, they write them in something else, and they compile them into JavaScript so that the browser will run them.

LISKOV: I mean, it's a sad statement about our field that a lot of big programs start off as small programs. And when you're building a small program, you don't have to pay much attention to methodology, because you can just do it through brute force.

BUTLER: Willpower.

LISKOV: Willpower. Well isn't brute force the same?

BUTLER: It's a form of willpower, right.

LISKOV: So this is why there's so much stuff written in languages that are not really all that great, because you start off this way, you don't have to think about it hard. Then when you get to the big problems, you're in trouble. But I don't know how to solve this.

BUTLER: Yeah, but we're OK for the reason that I said. JavaScript is a machine language. You can write your program in any language you want.

WARD: OK. I have a question addressed to Butler, Barbara, and Corby. And the question is this-- as computation becomes increasingly distributed, should we rethink the role of the operating system?

LISKOV: So there's a very healthy operating system community that thinks about this all the time. My basic belief is that distributed computing is probably the source of inspiration for parallel computing. Distributed computing, actually, is not in bad shape. And it is the way that Google does all its work today. It is the way that, in fact, companies that have lots and lots of processors-- those are parallel machines, in a sense. It's just that they use a particular memory model, and a particular way of thinking about how computation is organized.

BUTLER: Yeah, they run MapReduce.

LISKOV: Some of them run MapReduce. Some of them don't.

BUTLER: Most of them do, actually.

LISKOV: If you get down to the systems level, they're not running MapReduce. They're implementing MapReduce.

BUTLER: That is undeniably true. They're running some boring variant of Unix down at the system level.

LISKOV: Yeah, it's not clear to me that the operating systems are different. The problems that people look at in the operating system may change a little bit, the way you do load balancing, your notions of virtualization, they may evolve a bit. But it's not clear that they're very different than what they were.

BUTLER: Not much has changed in operating systems since the mid-70s. And in my opinion, that's not for commercial reasons. It's because we haven't had any great new ideas.

WARD: Aren't there people who argue that the web is now our operating system?

BUTLER: I have no idea what that means. Yes, certainly there are people who argue that, but I have no idea what it means.

WARD: OK. Here's a question, perhaps for our theorists. Has there been any progress made in the question of P=NP over the last 20 years? And do people still work on this problem?

YAO: I think that there are people who still work on this problem, as evidenced by the claim of solving P equal to NP, or the other way every few years. I think that the most recent one appeared with a very high publicity about a year ago. And so I think there are people who still work on it. And I think the last one got some. I think that most of the claims, people just brushed off. But somehow the last one got a little bit more esteem.

And so my feeling is that there has not been any real progress of P equal to NP. So I think that if you want to win a Turing Award by working on P equal to NP-- Butler, sorry-- I think that it probably is not a good idea for a PhD student.

BUTLER: I don't deny that.

RIVEST: We have made some progress, but of a negative sort. I mean, we do know why certain classes of arguments don't work. Some of the arguments relativize, or there's other issues with them. So we're getting more frustrated maybe. But we're understanding the narrow path we have to follow to come up with a proof, maybe.

WARD: OK. OK, let me ask you to turn up the gain on your crystal ball a lot. There will, presumably, be a celebration like this in 2061 of MIT's 200th anniversary. Will there be a Turing session? Will there be Turing Awards? Will computer science have just assimilated into other disciplines, like math and physics? Any ideas?

CORBATO: Well, I can speak up, and say that I don't expect to be there.

WARD: Well, we will certainly reserve a place for you, nonetheless.

YAO: Well, I have a feeling that at MIT 200, there was still be Turing Awards. And actually at that time, I think contrary to what you think, I think that computer science has absorbed physics, and biology, and everything else.

WARD: That's a great answer. OK but--

RIVEST: The person moderating the panel, Steve, will be a robot. It won't be a person.

WARD: Will the panelists be robots, as well?

RIVEST: We don't know.

WARD: Intriguing possibility. OK.

BUTLER: At ACM's 50th anniversary celebration, which was in 1997, my then boss, Nathan Myhrvold, made a prediction about what would be going on 50 years in the future. They asked a lot of people to do this. I would never have dared to make a prediction about what computing is going to be like in 50 years. But Nathan is fearless. He predicted uploading. So uploading means you get to upload your consciousness into a computer. Why would you want to do that? Well, you can be immortal.

WARD: Great.

BUTLER: And it seems to me, if you're a materialist, you can't really rule this out.

WARD: OK. Well, on that profound thought, I'd like to close this session by resurrecting an old tradition that Mike [INAUDIBLE] started many decades ago. He hosted the distinguished lecture series of LCS, which is now called the [INAUDIBLE] Lectures, and made a practice of asking each of his distinguished lecturers to summarize their life experience into a one-sentence lesson.

So I'm going to put each of you on the spot, and ask you for your one-sentence words of wisdom. And, of course, you're all Turing Award panelists. So you have the strong advantage that you can say something that's merely inscrutable, and the audience will interpret it as profound. So, Corby, can I ask you to start?

CORBATO: I'll give it a stab. I think if there's any single piece of advice one could give, it is to follow your enthusiasm, whatever that may turn out to be. And, in particular, take advantage of the ambience of a place like MIT, where one has to recognize in order to enjoy your life, you have to consider yourself a lifelong student. And you really should take a drink from the firehose.

WARD: Thank you, Corby. Butler?

BUTLER: I think Corby said it very well. You should do something that you really like doing. And if you like doing lots of things, I suggest that you can consider either biology, or chemistry, or computing as the hot fields for the next two or three decades, at least.

WARD: Good. Barbara?

LISKOV: So I think you should be opportunistic. I think opportunism is a good thing. And what it means is you look around, and you see what's moving and what's open. And then you should focus on what are the important problems, and avoid doing incremental work. And if it requires changing your field, then you shouldn't be afraid to do that.

WARD: Ron?

RIVEST: I guess it's very much in the same sort of spirit. The thing I would advise students would be to ask questions based on the hot technology of the day, but to ask them in such a way and try to derive answers that will be interesting a couple decades from now.

WARD: Good. Andy?

YAO: My advice for the students would be to always try to do something that is just a little bit beyond your ability.

WARD: Great. Well, I'd like to thank our distinguished panelists.

BUTLER: We have four and a half minutes.

WARD: Well, would you like to talk more about [INAUDIBLE]?

BUTLER: This is the first time I've ever seen one of these things cut off before its deadline.

WARD: I was going by my watch.

LISKOV: Oh. So there's--

RIVEST: Other questions?

WARD: Should we wing it, and take a question from the audience?

CORBATO: Sure.

LISKOV: Yeah.

[INTERPOSING VOICES]

AUDIENCE: On the P=NP problem, what about proving that neither P=NP nor P not equals NP is provable?

YAO: I would say, it's possible. But you may not be able to know whether that's true or not.

[INAUDIBLE]

AUDIENCE: Do you think you could entertain another question?

WARD: Sure. Stand close to the mic.

AUDIENCE: Sorry. Just wanted to make sure that I was allowed to. You were talking about the future. I was just wondering what you might think. One of you said machine learning was a very hot topic in the future. And I might have been busy and haven't heard the term IMB [INAUDIBLE] computer. And, of course, the fate of humanity as an [INAUDIBLE] search engine, because it can do more than what Google can do. It can take care of human language idiosyncrasies, and puns, and many other things. So I think that [INAUDIBLE] changes, it could be a stronger search engine than Google, and content search.

Also that it only runs at 80 [INAUDIBLE], which is a rather slow speed, because that speed has been around for a year or two. Maybe a year or so. Even [INAUDIBLE] specialized will probably be running faster floating point operations than that. Maybe their application is not a natural language.

The other thing I want to ask you about, if you want to comment on, besides the Watson computer's machine learning's success and searching engine success is what do you think of Google and YouTube broadcasted as a network, like MTV? The public doesn't like newspapers. The public likes to generate their own news, because they think the newspapers and broadcast news are a little bit corrupt, and have their own agenda. So if YouTube can broadcast and watch their own news, it will be kind of like MTV for the young, because YouTube used to be young, but now it's replaced by Facebook. So if it weren't cable TV, it could be [? that ?] MTV [? charged ?] Facebook, which is really YouTube without the pictures.

CORBATO: Was that a question?

RIVEST: There's a lot there in that question. I think my piece of advice would be to focus. [INAUDIBLE] good things [INAUDIBLE] narrowed down [INAUDIBLE].

WARD: Well, there is the whole YouTube revolution, where the citizenship of the browsers and so on are generally--

[INTERPOSING VOICES]

WARD: We retract your invitation. Sorry. But we are, in fact, now officially out of time. Let's thank the panelists.