The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
ASEE
Paper ID #41267

An Experience Report on Reducing Barriers by Removing Prerequisites for a CS 1 Introductory Programming Course

Dr. Udayan Das, Saint Mary's College of California

Udayan Das is an associate professor and program director in computer science. Dr. Das's main area of research is Technical Language Processing (TLP). Current NLP approaches and LLMs are inadequate to dealing with the complexity of technical text that needs to be reasoned on in such a manner that the accuracy of the automated reading can be relied upon and the cross-referentiality of technical documentation can be captured. His current research is focused on developing higher reliability Technical Language Models (TLMs) which are essentially knowledge-graph backed LLMs that can pinpoint where information was drawn from within a complex information environment. He also works toward improving CS education, broadening participation in computing, and incorporating ethics into CS education.

Christopher Isaac Fulton

An Experience Report on Reducing Barriers by Removing Prerequisites for a CS 1 Introductory Programming Course

Udayan Das†
Mathematics and Computer Science
Saint Mary's College of California
Moraga, CA
udd1@stmarys-ca.edu

Chris Fulton
School of Continuing and Professional Studies
Loyola University Chicago
Chicago, IL
cfulton@luc.edu

ABSTRACT

Introductory programming has evolved in many places to become a CS0 course, enabling students to get their feet wet with programming without completing significant math coursework. However, a survey conducted of CS programs shows that most CS1 programming courses that count towards an undergraduate CS degree continue to have a math or CS0 prerequisite. This experience report discusses the impact of removing the math prerequisite at an R2 university (Loyola University Chicago) and a liberal arts college (Saint Mary's College of California). Our experience shows that the removal of prerequisites, making the course readily available for those interested in pursuing CS, had no significant impact on student performance. Having minimal prerequisites has beneficial effects in terms of diversifying the CS student body as well as enabling students to begin CS coursework early, often in the first semester, potentially impacting persistence, but also enabling students to decide, early, if CS is right for them. Programs should evaluate what prior knowledge is required to be successful in a CS program. The high success rate of students of various backgrounds taking CS certificates and pursuing graduate school also shows that aggressive prerequisites may be functioning as barriers to entering CS programs. If we are serious about supporting diversity, we need to acknowledge the wide disparity in high school education nationwide and that prerequisites are perhaps functioning as a needless barrier. Where the CS0 course doesn't count towards a degree, or there isn't space for that requirement in the program, it is also worth considering whether the CS0 prerequisite is necessary.

KEYWORDS

CS 1, Introductory Programming, First programming course, Diversity, Barrier reduction, Broadening Participation in Computing (BPC)

1 Introduction

Prior to Fall 2018, the CS 1 introductory programming course at Loyola University Chicago had calculus as a prerequisite. The same was true of a similar course at a Saint Mary's College until Spring 2022. In this experience report, we share our experience of having removed the calculus prerequisite and requiring minimal prerequisites for the CS 1 programming course. Dr. Das was the program director at the Loyola when the change was put in place and has since moved to Saint Mary's College of California as of Fall 2021. Prof. Fulton is the current program director at Loyola who has left the minimal prerequisite requirement for the CS 1 course as is based upon their prior experience with similar requirements at a large public R1 university and a technical college offering associate degrees. Several major types of academic institutions are thus covered by our collective experience and demonstrates that most students are prepared to study programming at a CS 1 level right away provided there are appropriate teaching and learning methodologies applied. At Loyola and Saint Mary's College the change to the prerequisite is coupled with peer tutoring; active learning; project-based learning; and strong student support through academic advising, course touchpoints, and student success coaches.

Evolution. There is an evolution in progress as far as what prerequisites are required to begin a CS 1 programming course. As In a survey scan, we found that 30% of institutions have limited or no prerequisites[1] required for students starting a CS 1. This no prerequisite approach is consistent with our experience and the purpose of this experience report is to share information with those programs that still have prerequisites, especially calculus prerequisites, and to encourage them to reconsider those requirements.

Impact. There has been a significant increase in students with no calculus background opting to take the course and when students are taking the course. Both authors have strongly encouraged students taking the CS 1 course as early as possible and the now almost all students take the CS 1 course in the first 2 semesters, with the vast majority taking it in the first semester.[2] Although this paper does not focus on graduate students from non-CS backgrounds, it is worth noting that at the Loyola those students in the CS and CS-adjacent masters programs often take the CS 1 course as prep for beginning graduate study. There, as in a majority of graduate CS and CS-adjacent programs there is no explicit undergraduate calculus requirement to begin graduate study in CS. This is an important observation and worthy of reflection by all CS faculty as to the necessity of the prerequisites at the undergraduate level. We believe this also supports our observation that student motivation is the biggest factor in ensuring student success in the CS 1 programming course.

Saint Mary's College of California (SMC) is a Minority and Hispanic Serving Institution (MSI/HSI) and the student population at Loyola's School of Continuing and Professional Studies (SCPS) is majority women and over 35% minority. And supporting diversity and inclusion is a key principle at both institutions.

[1] We will use "no prerequisites" throughout this paper to mean no prerequisites other than high school completion or GED.
[2] Note that at Loyola the first semester is split into 2 8 week terms.

CS 1 programming topics. In this experience report we use Tew et al. [1], curriculum 2013, and curriculum 2023 beta [2] as guides for a set of topics common to a CS 1 programming course. In our survey scan, we found that most CS 1 programming courses cover all of these topics with the only main variance being in whether or not objects and recursion are introduced. Table 1 summarizes these general topic areas and we find no dependence of any of these topics on calculus.

Table 1: CS 1 topic areas

Variables
Data types
Console I/O
Expressions
Branching and selection
Loops
Functions
File I/O
Modules and libraries
Objects
Recursion

2 Background / related work

In a survey of students in an applied CS program students reported that motivation was the biggest success factor in academic and professional success (Virkki [3]). Herbert et al. found that improving student engagement early can also significantly impact both retention and student performance [4].

As noted in the next section, 30% of scanned programs now accept CS 0 as a prerequisite for beginning the CS 1 course. The impact of the CS 0 course on CS 1 has been studied quite extensively [1], [5], [6] and it is not a surprise that taking a CS 0 course, in college or in high school improves

student performance in the CS 1 course. Our concern is whether the CS 0 course is a necessary prerequisite, and whether, as in our case, the removal of the CS 0 prerequisite enables students more flexibility in taking upper division courses. We could also not conclusively answer, based on public information, whether the CS 0 counts for a degree. (The majority of scanned programs do not count CS 0 for the CS major.)

As in our experience, Doyle et al. [7] found that lowering mathematics requirements had no significant impact on the performance of students in CS 1. Pejcinovic et al. [8] found that having or not having calculus previously not only did not impact performance in algorithmic tasks but did not have much impact on engineering problem solving tasks either, suggesting that the problem solving skill gains from calculus may not directly translate into other domains, especially computational problem solving which is fundamentally different from mathematical problem solving.

The impact of taking calculus in high school as a predictor has been studied more extensively and having taking high school calculus, especially AP calculus predicted better performance in CS programs [6]. Our intention in this experience report is not to refute those claims but to present our experience in demonstrating that students can be successful in CS 1 without having taken calculus first. This is partially driven by the observation that there isn't a direct relationship between topics taught in a CS 1 programming course (table 1, [1]) and calculus and our intuition that having calculus in high school may be indicative of other success factors. The disparity in the availability of AP courses [9], [10] is a major concern for us, and should be for the larger community, in terms of addressing equity in CS education. While certain prior preparation is necessary for students to begin CS programs, if there is prior prep that we can compensate for through teaching and learning techniques in the introductory CS course sequence [11], [12] rather than using those as barriers-to-entry, then we believe that that is the way we should proceed. We subscribe to the idea that learning programming is easy [13] provided the academic environment supports that. When there is tremendous interest and extremely high enrollments in CS courses, there is a temptation to be more exclusive, but high interest could be an opportunity for addressing the gaps of the past.

Active learning [14], [15], peer learning and pair programming [16], [17], mastery learning [18], project-based learning [18]–[20], stronger student supports [12], availability of peer tutors [21], creativity and open-ended projects [20], [22], [23], and making course materials relevant to the student body [15], [24], [25] have all been shown to be successful approaches to improving overall performance and persistence in CS programs and often have impacts on supporting diversity [12], [24], [26].

3 Survey Scan Results

We conducted a survey of CS programs based on publicly available information for 50 programs in the US. These programs included many top CS programs as well as academic institutions of various types, including Carnegie Classified R1 and R2 universities, private and state universities, liberal arts colleges, as well as community colleges. 30% of CS1 courses at surveyed

programs have no prerequisites to start the course. (Note no prerequisites other than high school completion or GED.) Of the CS1 courses that have prerequisites the breakdown is as follows. calculus or precalculus is a requirement in 24%. Another 12% require calculus or precalculus as a corequisite. Other math such as college algebra or mathematical reasoning account for 12%. CS 0 is an accepted prerequisite for the CS1 course in 30% of institutions. Note that the numbers do not add up to 100 because the listed requirements sometimes overlap. Ex: at Loyola University Chicago, a student can start CS 1 having completed either a precalculus course or a CS 0 programming course. UC Berkley requires Math 1A which can be taken concurrently along with a 3 or above on AP CS A or equivalent, which for the purposes of our discussion is a CS 0 equivalency. Still others require calculus or college algebra. An exhaustive accounting of the specific requirements is beyond the scope of this discussion. The full list of scanned programs and their requirements is available from the authors upon request.

Several programs from highly ranked CS programs have moved to the no prerequisite model including University of Illinois (Urbana-Champaign and Chicago), Harvard, Cornell, Virginia Tech, Duke, and Georgia Tech. Other major CS programs opt to start students at CS 0, including Stanford, University of Washington at Seattle, University of Wisconsin at Madison, the University of Southern California, University of California (multiple), and Caltech. However, there are still many others that require calculus, such as the University of Maryland College Park and the University of Texas at Austin. Cornell's CS Engineering degree requires a calculus corequisite. Princeton requires a CS0 course along with college-level Science. Surveyed community colleges require college algebra, with the exception of Harper College in Illinois which requires a CS 0 course in addition to college algebra.

Table 2: Scan Results of CS programs scan

CS1 course Prerequisite	Percentage
None	30%
Calculus or precalculus	24%
Calculus or precalculus co-requisite	12%
College algebra or mathematical reasoning	12%
CS0	30%

* Note: numbers do not add up to 100 because listed requirements sometimes overlap.

One curiosity, based on our reading of the publicly available requirements, is that in many cases the CS 0 courses do not count as part of the CS major (the University of Southern California is

a notable exception). We hope that the CS 0 does count towards the undergraduate degree. If CS 0 does not count in any way towards a student's program—as part of the major or as an elective—then this is a disservice to those students who may not have access to CS education or AP courses in high school. Given our experience, the CS 0 is not necessary for students of virtually any background to begin a CS1 course and therefore if CS 0 is a course that does not count for an undergraduate degree, then this is an additional burden for already disadvantaged students.

Overall, our survey indicates that there is an evolution in progress as far as what prior preparation is required for beginning a CS 1 course and by extension a CS major. We think therefore that this experience report is a valuable addition to the overall conversation and our experience of observing no impact of removing the existing math prerequisites is more data in favor of letting students regardless of their prior preparation to begin CS studies immediately, which we expect has considerable benefits. We discuss our experiences more directly in the reflections section below.

4 Instructor and Program Director Reflections

Between the 2 authors we have several years of collective experience of teaching CS1 with no prerequisites other than high school completion or GED, and overseeing programs that have those CS1 courses in the CS or applied CS major. We have served as instructors of the CS1 programming courses, and we are both Program Directors overseeing our respective programs.

At the liberal arts college, the CS1 requirement is part of BS and BA CS majors. At Loyola, the CS1 requirement applies in a continuing studies program where students primarily come for degree completion. This has implications for student motivations which are discussed in a reflection below. Loyola a traditional undergraduate program still exists which has more traditional prerequisites. Further, in the continuing studies context, the CS1 course is offered in an accelerated 8-week online format, which would in theory present more challenges for students who have little to no background in programming. While it may be beneficial to add a CS0 course, in the context of both institutions the addition of the extra course would be significantly challenging. And, based on our experience, and this experience report, we expect that there is more value added in additional upper division electives than adding the CS0 course into our respective programs.

4.1 Instructor 1 Reflection

I began working at Loyola in January 2018 and continued until the end of June 2021. I was in charge of teaching coursework and overseeing programs in the continuing studies school. There was a BA Information Technology which was an applied CS degree as well as a CS certificate. The student population at this continuing studies school is primarily adult learners looking to complete college as well as those seeking career change and upskilling. My experiences running a non-profit training program for underserved individuals prior to starting my position at Loyola had taught me that those with little or no background could be brought up to speed as far as programming is

concerned with the right teaching and learning approaches bolstered by the appropriate student supports. At the same time, there were challenges with students being hesitant to take calculus simply to be able to take the CS 1 introductory programming course. For those seeking a CS certificate the additional course would also essentially increase their time-to-completion and cost-of-completion by a third. Both BAIT and CS certificate students were also likely to have been away from schooling for a while and starting schooling again with calculus was a daunting prospect. Taking everything into account, and building on my non-profit experience, I decided to remove the calculus prerequisite. Given the standard CS 1 curriculum I did not see any dependence on calculus and thought that a calculus requirement may have a gate-keeping function than impacting learning in a CS 1 course.

In Fall 2021, coming to SMC as a CS program director in a Mathematics and CS department I once again found that there was a historical calculus prerequisite for the CS 1 intro programming course and successfully had that requirement removed applicable Spring 2022. In all, I have seen 9 semesters of teaching of a CS 1 course with minimal requirements and found that it has no impact on student performance. In fact, over time the traditional CS department has begun sending graduate students with non-CS programs to the continuing studies courses so that those students can meet the prerequisites needed to begin graduate school. This is unsurprising since CS certificate students of various backgrounds have gone on to top 10 graduate programs indicating the quality of their education and the fact that students were able to meet the learning outcomes needed to begin graduate school in CS. (Incidentally, it is worth noting that graduate CS programs have less stringent math requirements than undergraduate CS programs which we find interesting.)

I have, however, observed several benefits of removing the prerequisite. At Loyola, students previously turned away from both the BAIT and CS certificate due to the calculus requirement were now able to pursue the degree. Students from other majors such as Management and Psychology were able to complete the CS certificate as an added credential. At SMC, a wider range of students and those interested in CS have been able to take the CS 1 course. An English major who became interested in CS after taking a programming workshop over Summer 2022 was able to take CS 1 immediately in Fall 2022 and has since become a double major in English and CS (pursuing the non-calculus BA CS) which would not be possible if the calculus was a prerequisite for the CS 1 course and a non-calculus BA CS was available. I believe one of the greatest drivers of student success in the CS 1 course is motivation and starting students as early as possible in what got them interested in the discipline in the first place can go a long way toward maintaining motivation. As noted earlier, motivation was also self-identified by surveyed students as being the biggest success factor in Virkki [3].

We offer a CS 0 course meant to introduce students to computing and computational thinking which is also proving quite popular among non-CS majors. Many non-CS-majors such as those in psychology or sociology benefit greatly from having programming skills. Additionally, those students now also have the opportunity to pursue the 3 course CS certificate or the 6 course CS

minor neither of which require calculus making it more accessible to students of different backgrounds.

When there is a CS 1 class with students who bring a wide range of prior preparation to the course there is a course management challenge, and it is incumbent upon the instructor to ensure that there are support mechanisms for students who may be struggling with the course materials. In addition to the training of peer tutors and TAs, in my case this has meant ensuring that whenever I am teaching CS 1, I have accounted for extra time needed for the course. In the CS 1 course students are required to meet with me 1-on-1 which helps me assess student progress as well as strategize, often with student agency, how to help each student succeed. Active learning and peer learning techniques are also critical to ensuring there is a greater opportunity for students to learn from each other and form a learning community. In the future, I also plan to experiment with a pass/fail and/or mastery learning approach for the base requirements of the course.

4.2 Instructor 2 Reflection

Starting in Fall 2022, I began teaching at Loyola with a primary audience of adult learners through the school of continuing studies. Before my current role, I held roles at both an R1 institution and a small private institution. The latter was focused on expediting students into the workforce by offering associate degrees centered around computer science. Neither my current institution nor the ones I previously mentioned required prerequisite for CS 1. Through a variety of institutional contexts, I was able to observe students succeeding in CS 1 without having taken the prerequisites that are widely required in CS programs.

A considerable number of my students have been adult learners. Adult learners have expressed that barriers for returning to school are higher in comparison to that of a traditional student transitioning from high school. Factors such as gathering admission material to adopting new life routines in preparation for pursing education are major adjustments and having to take courses that do not particularly align with career interest influenced their choice on what programs would suite their career interest. It is often the excitement and drive to immediately start learning a technical skill that motivates students to return to school. Immediately capturing the interest of what drew a student back to school is what resulted in higher engagement and matriculation. What I observed was without the immediate gratification of learning that ultimately brought them back to school, the requirement of taking other courses before a CS 1 course deterred students from enrolling in a CS 1 course due to the misconception that programming involved a significant amount of math. I found this to be the case when students were encouraged to complete general education courses that involved math prior to enrolling in technical courses. Removing prequisites to a CS 1 course can attract more returning students, without imposing a barrier of advanced mathematics such as calculus, which may have little relevance to their interests or to the actual course material.

While reflecting on the jobs students obtain after graduation, many of the technical jobs and skills needed are not predicated on having prerequisites such as calculus. A good portion of the jobs

students obtained included skills such as creative thinking, programming, and problem-solving skills, but most did not appear to require an advanced level of mathematics such as calculus. I do not advocate for the elimination of courses like calculus, but rather their exclusion as entry criteria for enrolling in a CS 1 course. During my initial teaching role at the private institution, students had the option to complete a 2 + 2 program: 2 years for an associate degree and the remaining 2 years for the bachelor's degree completion. The associate degree included a mixture of general education and technical courses to prepare students for entry-level tech jobs. The subsequent two years focused more on theory-based and advanced mathematics courses, including calculus. Once a student obtained an associate degree, they were encouraged to pursue employment and gain meaningful work experience before returning to complete the bachelor's degree. There were a significant number of benefits to this approach, some included students having a keen awareness of how material would apply to real world scenarios, employers offered students tuition reimbursement to offset cost, students had a clear objective for how their education would supplement their career pursuits. The courses students completed upon returning to earn the bachelor's included calculus and other theory-based courses that provided more relevance to a deeper understanding of concepts relayed. Students also took more thoughtful consideration of what courses would be applicable to their career objective and engaged more in the advising process taking more agency over their programs of study.

During my time at the R1 institution, the absence of prerequisites for CS 1 led to diverse students from various disciplines enrolling in the course which resulted in high enrollment. The accessibility of the course to non-computer science majors broadened the reach to majors outside of the school who found value in the course. Students from the school of design expressed the relevance of the CS 1 material when developing creative digital art. I observed students from other disciplines continuing in subsequent courses with successful completion of CS 1 due to finding an interest in programming. Not having prerequisites for CS 1 allowed administrators to easily approve a request for students who desired to enroll in the course as a general elective if there was space.

4.3 Pedagogical techniques and student support

In this section we present some of the teaching techniques applied in courses in our respective classes and programs. We present this in the interest of giving a complete understanding of all factors that influence the success of students at Loyola SCPS and SMC.

At a classroom level, active learning techniques are used extensively, including liberal use of groupwork. Project based learning is used throughout the intro programming courses at both institutions. Das uses an open ended final project in the intro programming course called "Bring your own project" (BYOP) which definitely drives student engagement [20]. We both heavily rely on online materials such as Wiley Zybooks [27] as an active textbook environment.

Additional support measures and techniques involve creating peer-to-peer learning opportunities and providing students the opportunity to receive individual assistance. Fulton schedules designated in-class times for students to work together on coding activities. These activities provide an opportunity for students to process the material with others who vary on the spectrum of knowledge. Guidelines are provided for peer-to-peer opportunities which encourage students to ask questions and to share their understanding of the code and share helpful information that contributed to their overall understanding.

5 Discussion

We will focus this discussion on the following questions. What is the rationale behind the calculus prerequisite for a CS 1 programming course? How does a CS 0 prerequisite compare with the calculus prerequisite? What is the result of removing either a calculus or CS 0 prerequisite?

As discussed in the introduction, the typical content for a CS 1 introductory programming course does not have any dependence on calculus. The standard topics are usually composed of variables and expressions, I/O including working with files, branching, loops, functions, and objects. None of these topics require any knowledge of calculus, and high school level math is sufficient for embarking on learning any of these topics. Thus, we strongly advocate for a reconsideration of calculus prerequisite. A calculus prerequisite can function as a barrier for students who have had less advanced math preparation in the past, which includes access to AP coursework. The calculus prerequisite is a significant self-selector towards which students are able to begin the CS 1 programming course and when. Students who are able to begin programming courses sooner have an advantage over students who take that course later, further amplifying pre-college disparities. At the same time, in our experience, the mere presence of the calculus prerequisite discourages many students from even considering taking the CS 1 course. The main advantage of taking calculus should be in learning problem-solving skills. But computational problem solving is different than mathematical problem-solving. Authors such as Pejcinovic et al. [8] found that calculus not only had an inconclusive effect in improving computational problem-solving but general engineering problem-solving as well.

The CS 0 prerequisite is more in line with the needs of a CS 1 course. The CS 0 course is also well positioned as a trial course for students considering CS as an option. However, in our experience the CS 0 course is not necessary for success in the CS 1 course. Whether the CS 0 course should exist should be dependent upon whether or not the course counts towards the degree as a whole. In our scan we were not able to definitively answer whether all programs that require a CS 0 prerequisite count that course towards a degree. Programs that do not count the CS 0 course as part of the major hopefully still count the course as part of the degree. If they do not, then this is a needless burden for those students who did not have access to a CS 0 course previously (AP or otherwise). On the other hand, in the case of our programs with limited size majors, the lack of a CS 0 course allows for the inclusion of upper division courses.

This experience report indicates that there is no difference in student performance between CS 1 programming course requiring a calculus prerequisite and one without that requirement. Granted that there are other factors such as the instruction methodology and the student support that impact this result. However, if we are serious as a community regarding broadening participation in computing it is incumbent upon us to consider whether the prerequisite is necessary or is functioning as a barrier to students starting the CS 1 programming course, and by extension starting CS programs.

While prior experience [5] and having some form of high school calculus, especially AP calculus, has been shown to predict improved student success through an introductory CS sequence, towards broadening participation in computing we should as a faculty community be working towards teaching and learning approaches that bridge the prior experience gap. We also note that there is no strong counter-factual since not having prior math preparation does not predict failure in the intro programming course. We should be concerned with who has access to AP courses [9], [10], [29] and whether or not the erecting of barriers based on high school access or privilege is justified.

The long-term performance of our students at Loyola and Saint Mary's also indicates that students who have taken these CS 1 classes have continued to be successful. This should suggest reconsideration of when and where in the curriculum the math and/or calculus requirement should show up. Can we change the function of the math/calculus requirement from a gatekeeper to a when needed requirement. Note that while the BS CS at Saint Mary's includes Calculus I and Calculus II, the reason for their inclusion is to mainly support understanding of the underbelly of Machine Learning and Artificial Intelligence. The BA CS at Saint Mary's and the BA IT at Loyola do not have a Calculus requirement.

We reiterate what was noted earlier that Calculus is often not a requirement for entry into graduate level CS programs. Those coming from non-CS backgrounds are asked to take prerequisite courses that are often some type of accelerated intro to computing and/or coursework in programming, data structures, and algorithms, but not Calculus.

It is time to ask what the utility of the Calculus course is to a Computer Science major? Colleagues mention the need to learn problem solving techniques. In our experience, the techniques needed to successfully solve computing problems are different than calculus. This is not to say that knowing Calculus is not beneficial; undoubtedly having Calculus under their belt is greatly beneficial particularly in ML/AI and Engineering oriented applications. We look forward to animated discussions on this topic.

6 Conclusions and future work

The removal of prerequisites has significant benefits in terms of allowing a wider variety of students to take the CS 1 course earlier in their programs. The rationale of requiring students to

take calculus prior to taking a CS 1 course should be reconsidered. Not only is there no dependence of a standard CS 1 programming course on calculus, but this requirement can also function as discouragement and a delay in students interested in taking the CS 1 programming course. Towards broadening participation in computing and supporting students from a wide variety of backgrounds, bringing differing prior experience a big step would be making it easier for students to begin programming as soon as possible. As recognized in the ACM/IEEE/AAAI CS 2023 curriculum guide beta, introductory programming topics and knowledge areas do not have required math prerequisites. The results of our scan indicate that a variety of institutions (30%) have come to the same conclusion as this experience report and enable students to begin the CS 1 programming course very early in the CS programs.

We reiterate that we are not calling for a removal of calculus from CS or CS-adjacent programs. That depends upon the type of program under consideration. However, we strongly advocate for the removal of calculus as an entry criterion into the CS 1 programming course, thereby reducing a barrier-to-entry.

For future work, we would like to expand the scan to include more institutions and make that information available online and allow institutions to update and comment on their thought process behind the prerequisites for the CS 1 course. It would also be interesting to see how the requirements have evolved over time by reviewing older course catalogs. Once the CS2023 curriculum guide is finalized, we would also like to repeat the survey scan and analysis in 2 years to evaluate how programs are responding to the new guide.

ACKNOWLEDGMENTS

We would like to thank the Computer Science department chair and Dean at the Loyola College of Arts and Science as well as the Interim, Associate and Assistant Deans of the School of Continuing and Professional Studies at the Loyola University Chicago. We would also like to thank the Mathematics and Computer Science department chair, and the Dean of the School of Science at Saint Mary's College of California. As ever, we are of course eternally thankful and grateful for our wonderful students.

REFERENCES

[1] A. E. Tew and M. Guzdial, "Developing a validated assessment of fundamental CS1 concepts," in *Proceedings of the 41st ACM technical symposium on Computer science education*, Milwaukee Wisconsin USA: ACM, Mar. 2010, pp. 97–101. doi: 10.1145/1734263.1734297.

[2] "CS2023 – ACM/IEEE-CS/AAAI Computer Science Curricula." Accessed: Aug. 18, 2023. [Online]. Available: https://csed.acm.org/

[3] O. T. Virkki, "Performance and Attrition in Information Technology Studies; A Survey of Students' Viewpoints," in *2023 IEEE Global Engineering Education Conference (EDUCON)*, May 2023, pp. 1–9. doi: 10.1109/EDUCON54358.2023.10125231.

[4] N. Herbert, "Impact of Student Engagement on First Year ICT Performance," in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2017, pp. 1085–1090. doi: 10.1109/CSCI.2017.189.

[5] G. Bui, N. Sibia, A. Zavaleta Bernuy, M. Liut, and A. Petersen, "Prior Programming Experience: A Persistent Performance Gap in CS1 and CS2," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, Toronto ON Canada: ACM, Mar. 2023, pp. 889–895. doi: 10.1145/3545945.3569752.

[6] C. Chen, J. M. Kang, G. Sonnert, and P. M. Sadler, "High School Calculus and Computer Science Course Taking as Predictors of Success in Introductory College Computer Science," *ACM Trans. Comput. Educ.*, vol. 21, no. 1, pp. 1–21, Mar. 2021, doi: 10.1145/3433169.

[7] M. Doyle, D. Kasturiratna, B. D. Richardson, and S. W. Soled, "Computer Science and Computer Information Technology majors together: Analyzing factors impacting students' success in introductory programming," in *2009 39th IEEE Frontiers in Education Conference*, Oct. 2009, pp. 1–6. doi: 10.1109/FIE.2009.5350582.

[8] B. Pejcinovic, M. Holtzman, P. K. Wong, and G. Recktenwald, "Assessing student preparedness for introductory engineering and programming courses," in *2017 IEEE Frontiers in Education Conference (FIE)*, Oct. 2017, pp. 1–5. doi: 10.1109/FIE.2017.8190539.

[9] J. R. Thomas, "Access to AP courses often elusive for low-income students," CT Mirror. Accessed: Aug. 17, 2023. [Online]. Available: http://ctmirror.org/2018/05/14/advanced-placement-debate-open-closed-gate/

[10] G. Siegel-Hawley, K. Taylor, E. Frankenburg, and K. Bridges, "Segregation within Schools: Unequal Access to AP Courses by Race and Economic Status in Virginia," Apr. 2015.

[11] M. S. Kirkpatrick and C. Mayfield, "Evaluating an Alternative CS1 for Students with Prior Programming Experience," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, Seattle Washington USA: ACM, Mar. 2017, pp. 333–338. doi: 10.1145/3017680.3017759.

[12] P. Dempster, D. Onah, and L. Blair, "Increasing academic diversity and inter-disciplinarity of Computer Science in Higher Education," in *Proceedings of the 4th Conference on Computing Education Practice 2020*, Durham United Kingdom: ACM, Jan. 2020, pp. 1–4. doi: 10.1145/3372356.3372366.

[13] A. Luxton-Reilly, "Learning to Program is Easy," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, Arequipa Peru: ACM, Jul. 2016, pp. 284–289. doi: 10.1145/2899415.2899432.

[14] N. Dehbozorgi, "Active Learning Design Patterns for CS Education," in *Proceedings of the 2017 ACM Conference on International Computing Education Research*, in ICER '17. New York, NY, USA:

Association for Computing Machinery, Aug. 2017, pp. 291–292. doi: 10.1145/3105726.3105741.
[15] E. F. Gehringer and C. S. Miller, "Student-generated active-learning exercises," in *Proceedings of the 40th ACM technical symposium on Computer science education*, in SIGCSE '09. New York, NY, USA: Association for Computing Machinery, Mar. 2009, pp. 81–85. doi: 10.1145/1508865.1508897.
[16] L. Porter and B. Simon, "Retaining nearly one-third more majors with a trio of instructional best practices in CS1," in *Proceeding of the 44th ACM technical symposium on Computer science education*, Denver Colorado USA: ACM, Mar. 2013, pp. 165–170. doi: 10.1145/2445196.2445248.
[17] J. Brougham, S. Freeman, and B. Jaeger, "Pair Programming: More Learning And Less Anxiety In A First Programming Course," in *2003 Annual Conference Proceedings*, Nashville, Tennessee: ASEE Conferences, Jun. 2003, p. 8.912.1-8.912.9. doi: 10.18260/1-2--11728.
[18] M. Jazayeri, "Combining Mastery Learning with Project-Based Learning in a First Programming Course: An Experience Report," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Florence, Italy: IEEE, May 2015, pp. 315–318. doi: 10.1109/ICSE.2015.163.
[19] M. Frydenberg and K. Mentzer, "From Engagement to Empowerment: Project-Based Learning in Python Coding Courses," 2021.
[20] U. Das, "BYOP: 'Bring Your Own Project': How student-driven programming projects in an introductory programming course can drive engagement and continuous learning," presented at the 2023 ASEE Annual Conference & Exposition, Jun. 2023. Accessed: Aug. 18, 2023. [Online]. Available: https://peer.asee.org/byop-bring-your-own-project-how-student-driven-programming-projects-in-an-introductory-programming-course-can-drive-engagement-and-continuous-learning
[21] J. A. Cottam, S. Menzel, and J. Greenblatt, "Tutoring for retention," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, Dallas TX USA: ACM, Mar. 2011, pp. 213–218. doi: 10.1145/1953163.1953227.
[22] T. VanDeGrift, "Supporting Creativity and User Interaction in CS 1 Homework Assignments," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, in SIGCSE '15. New York, NY, USA: Association for Computing Machinery, Feb. 2015, pp. 54–59. doi: 10.1145/2676723.2677250.
[23] S. Sharmin, "Creativity in CS1: A Literature Review," *ACM Trans. Comput. Educ.*, vol. 22, no. 2, p. 16:1-16:26, Nov. 2021, doi: 10.1145/3459995.
[24] R. Varma, "Making computer science minority-friendly," *Commun. ACM*, vol. 49, no. 2, pp. 129–134, Feb. 2006, doi: 10.1145/1113034.1113041.
[25] B. Hui, P. Rajabi, and A. Pinchbeck, "Disparity Between Textbook Examples and What Young Students Find Interesting," in *2021 IEEE Frontiers in Education Conference (FIE)*, Oct. 2021, pp. 1–8. doi: 10.1109/FIE49875.2021.9637145.
[26] W. Wei, J. J. Ryoo, and A. Morris, "Centering Minoritized Students' Perspectives: What Makes CS Learning Consequential," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, in SIGCSE 2023. New York, NY, USA: Association for Computing Machinery, Mar. 2023, pp. 666–672. doi: 10.1145/3545945.3569878.
[27] "zyBooks - Build Confidence and Save Time With Interactive Textbooks," zyBooks. Accessed: Jan. 09, 2024. [Online]. Available: https://www.zybooks.com/home/

[28] C. F. Reilly and E. Tomai, “An examination of mathematics preparation for and progress through three introductory computer science courses,” in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, Oct. 2014, pp. 1–9. doi: 10.1109/FIE.2014.7044349.
[29] “AP versus the excellence gap,” The Thomas B. Fordham Institute. Accessed: Aug. 17, 2023. [Online]. Available: https://fordhaminstitute.org/national/commentary/ap-versus-excellence-gap