Lightning Talks of EduHPC 2020

Joel C. Adams* Godmar Back** Piotr Bała‖ Michael K. Bane¶ Kirk Cameron**
Henri Casanova† Margaret Ellis** Rafael Ferreira da Silva‡, Gautam Jethwani‡,
William Koch†, Tabitha Lee§, Tongyu Zhu‡,
*Department of Computer Science, Calvin University, USA
†Information and Computer Sciences, University of Hawaii, USA
‡Information Sciences Institute, University of Southern California, USA
§Electrical Engineering and Computer Science Department, Vanderbilt University, USA
¶ Department of Computer Science, University of Liverpool, United Kingdom
‖ Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Poland
** Department of Computer Science, Virginia Tech, USA

***Abstract*—Lightning talks of EduHPC are a venue where HPC educators discuss work in progress. This paper summarizes the EduHPC 2020 lightning talks, which cover four very different areas: (i) The simulation-based pedagogy of the EduWRENCH project, including motivations for using simulation to teach High Performance Computing, the design principles underlying EduWRENCH modules, a survey of the available modules, a look at a particular module, plus a conclusion including lesson learned thus far and future plans. (ii) The use of the software-tuning component from Student Cluster Competitions in the HPC master's program at the University of Liverpool. (iii) Steps being taken by the Computer Systems Genome Project at Virginia Tech to foster a community atmosphere among the diverse students working to catalog the lineage of computer system performance over time. (iv) A 3-semester master's degree program titled Computational Engineering, focused on HPC training, being offered at the University of Warsaw.**

***Index Terms*—computer science education, high performance computing education, education tools, master's curriculum**

I. Introduction

High Performance Computing (HPC) and more generally Parallel and Distributed Computing (PDC) have become pervasive, from supercomputers and server farms containing multicore CPUs and GPUs, to individual PCs, laptops, and mobile devices. Even casual users of computers now depend on the parallel and distributed capabilities inherent in their smartphones, tablets, and/or laptops.

With HPC and PDC potentially available throughout the computational ecosystem, it is important for every computing student and faculty member understand how parallelism and distributed computing impact computational problem solving. It is essential for educators to impart a range of PDC and HPC knowledge and skills at multiple levels within the computing curriculum. However, rapid changes in hardware platforms, programming languages, and software development environments make it increasingly challenging for educators to decide what to teach and how to teach it.

In this context, EduHPC, the Workshop on Education for High-Performance Computing, has positioned itself as the primary venue for educators to share and discuss innovative ideas to enhance HPC education. In particular, EduHPC's Lightning Talks provides a venue where educators can present short talks on works in progress, methods that have not been fully tested, or even just ideas for yet-untried innovations, to garner reactions from the community. This paper provides overviews of the four Lightning Talks from EduHPC 2020:

1) A look at the EduWRENCH Project, which emphasizes simulation-based pedagogy. This talk includes motivations for using simulation to teach High Performance Computing, the design principles underlying EduWRENCH modules, a survey of the available modules, a look at a particular module, plus a conclusion including lesson learned thus far and future plans.
2) An overview of how the Department of Computer Science at the University of Liverpool has incorporated the aspects of the SC and ISC Student Cluster Competitions into their HPC master's program, specifically the project of making a given software package run as fast as possible on any of their available hardware platforms.
3) Concrete steps being taken to foster a supportive community atmosphere among the diverse students working to catalog the lineage of computer system performance over time within the Computer Systems Genome (CSGenome) Project at Virginia Tech. This talk addresses specific team culture, management, and communication strategies that have proven successful at broadening student participation in their project.
4) An overview of the master's degree program in Computational Engineering from the Interdisciplinary Centre for Mathematical and Computational Modeling at the University of Warsaw. This program's aim is to train HPC users and administrators; the talk describes the program's goals, its curriculum for achieving those goals, and an analysis of the program's achievements of those goals after 4 years of experience.

Section II describes the EduWRENCH talk, Section III describes the X talk, Section IV describes the CSGenome talk, and Section V describes the Warsaw master's degree talk.

II. EduWRENCH: Simulation-Driven Pedagogic Modules

By: William Koch, Tongyu Zhu, Gautam Jethwani, Tabitha Lee, Rafael Ferreira da Silva, Henri Casanova

Teaching PDC and HPC concepts is challenging. Teaching purely "on the blackboard" is often not very compelling and does not allow students to learn as effectively as they could if provided with hands-on learning opportunities. In practice, hands-on teaching means having students develop programs and run them on hardware/software platforms that are provided to them. This approach, unfortunately, faces two kinds of challenges. First, there are *participation challenges*. It is well documented that providing students with usable and representative hardware/software platforms is not always feasible, at least not at all institutions. Furthermore, having students develop actual programs adds several prerequisites to teaching PDC and HPC concepts (e.g., students need to know how to write multi-threaded programs, students need to know how to interact with a cloud infrastructure or a batch scheduler). As a result, these concepts cannot be taught early on in the curriculum, or at least not in a hands-on manner. Second, there are *pedagogic challenges*. Students are only exposed to those platform configurations made available to them, which precludes achieving several learning objectives (e.g., understanding how different network configurations impact parallel program performance). Also, students need to be trained in platform usage mechanisms and policies, which can take away too much time away from other learning objectives, especially early in the curriculum. Finally, making sure that students are exposed to all relevant concepts via programming assignments is often not feasible, or would require an enormous amount of work on behalf of the instructors and of the students.

Addressing both kinds of challenges above would mean providing students with hands-on learning opportunities for achieving PDC and HPC learning objectives effectively, without having them develop and run code. One approach is to rely on *simulation*, i.e., simulate parallel application executions using a software artifact that mimics real-world executions. Using simulation there is no need for actual platforms and students can run experiments easily on their own computer or in the browser. In addition, arbitrary platform configurations can be simulated. Finally, a number of modalities can be used in terms of the level of details exposed to the students, from having students simply run "canned" simulated executions to having students develop programs and then simulate their execution.

In the next section, we describe the recently developed EduWRENCH pedagogic modules, which target PDC and HPC learning objectives. The most important aspect of these modules is that they include simulation-driven activities that provide students with hands-on learning opportunities. These modules have few prerequisites and can be completed in sequence by independent learners. But they are also designed to be easy to integrate piecemeal into existing university courses. In this case, the goal is to target learning objectives that are not currently targeted, or to complement existing course content. The modules available to date do do not require any programming by the students, and thus can be used early in the Computer Science curriculum. They have already been used effectively in existing undergraduate courses courses to complement course content. Most recently, in the Fall 2020 semester, some of these modules are being used to enhance the content of a programming-heavy, upper-division course on concurrent and high performance computing at the University of Hawai'i at Mānoa. Although many learning objectives are targeted by the programming assignments in that course, several relevant learning objectives are not. The use of the EduWRENCH modules allow students to achieve these learning objectives in a hands-on manner, i.e., using simulation.

A. The EduWRENCH Modules

At the time of writing, the EduWRENCH modules are publicly available at https://eduwrench.org and cover the following broad PDC and HPC topics:

- Single-core computing (work and compute speed, time-sharing, RAM constraints, I/O operations);
- Multi-core computing (task parallelism, load balancing and idle time, task dependencies, data parallelism);
- Networking (latency, bandwidth, topology, contention);
- The client-server model (concepts, pipelining of communication and computation);
- The Coordinator-worker model (concepts, scheduling);
- The Workflow model (concepts, locality, mixed parallelism).

Each module is a single Web page with multiple tabs, where each tab targets a particular topic relevant to the topic. Each tab contains:

- A pedagogic narrative;
- In-the-browser, interactive, simulation-driven activities;
- Practice questions with revealable solutions;
- Open questions without solutions that can be used by an instructor as homework assignments.

Several practice and open questions explicitly rely on the simulation-driven activities. Some modules have a last "capstone" tab in which students go through case-studies in which they apply all the concepts they have learned in the module. These case-studies can also be used as self-contained homework assignments by an instructor. The simulation activities rely on simulators implemented in C++ using the WRENCH simulation framework, which itself builds on the lower-level SimGrid simulation framework.

Overall, the available modules at the time this is written comprise 86 practice questions, 92 open questions, and 16 simulation-driven activities.

B. Classroom Use

Preliminary versions of the EduWRENCH modules were used in the classroom in Spring'19 and Fall'19 offerings of a 300-level undergraduate courses at the University of

Hawai'i (Operating Systems). Some modules were completed by students independently, some modules were covered in interactive in-class sessions with students working in teams and the instructor providing scaffolding, part of a module was used as a homework assignment, and the final exam included assessment questions for the learning objectives targeted by all these modules.

In both semesters quantitative and qualitative data was collected about the student experience: pre- and post-knowledge tests, student feedback via questionnaires asking about overall experience and self-perceived learning, logs of student use of simulation-driven activities, and assessment of learning via homework assignments and exam questions. This data show that most students were engaged in the simulation activities, with most student in the class running a number of simulations in line with what was expected, or larger. Another finding is that there is positive correlation between the number of simulations students run and their grades on relevant exam questions, even though some students who ran very large numbers of simulations did poorly on the questions. We surmise that these students struggled with the material and ran many simulations haphazardly. Finally, feedback from students shows that they had a very positive experience. For instance, in the Fall'19 semester, 71% of students deemed the material "very useful", 29% "somewhat useful", and 0% "not useful". 87% of students answered "yes" to the question "Are you interested in learning more about Parallel and Distributed Computing?". Most importantly, when asked how useful was the use of simulation for learning, 81% deemed it "very useful" and 19% "somewhat useful". Written-in comments by students that semester included: "I liked the simulation. It was a nice addition to visually see as well as check my work", "I like the hands-on experience during class and having it available at any time", "Love the visuals". Overall, these preliminary evaluation results were conclusive regarding the effectiveness (actual or self-perceived) of the preliminary pedagogic modules.

C. Key takeaways

Based on the experience and student feedback in these two semesters, we made a number of changes/improvements to the pedagogic content. Namely, we split several modules into sub-modules, added practice questions, and integrated the simulation-driven activities in the same page as the pedagogic narrative (as opposed to it being executed in a different browser tab). These improvements have lead to the EduWRENCH modules currently available at https://eduwrench.org. These newly available modules have been used at the University of Southern California to train individual students before they participate in parallel computing and high performance computing research projects. They are also being used in yet another undergraduate course at Univ. of Hawai'i in the Fall 2020 semester.

Besides disseminating the existing modules to instructors, future work entails adding new content to existing modules and developing new sets of modules. For instance, we are currently creating modules that focus on concepts and practice of Cyberinfrastructure Computing for scientific applications, and modules are being developed for teaching students the concepts behind batch scheduling, and for practicing, using and optimizing the use of a batch scheduler, all in simulation.

Acknowledgments. This work is funded by NSF contracts #1923539 and #1923621: "CyberTraining: Implementation: Small: Integrating core CI literacy and skills into university curricula via simulation-driven activities"; and partly funded by NSF contract #1659886.

III. Evolving the traditional Student Cluster Competition as tomorrow's "Peachy Assignments"

By: Michael K. Bane

Motivating the Assignment

At University of Liverpool, the MSc course on HPC was revamped during the academic year 2018-2019. Additionally a 3rd year undergraduate course on HPC was designed and introduced for the academic year 2019-2020. An underlying principle of the new teaching approach has been "authenticity", in line with the University's guiding principles via Curriculum 2021 [1]. Assignments had been aligned with specific learning objectives which had been on specifics such as OpenMP for shared memory programming and MPI for distributed memory programming. The new undergraduate course, running whilst the covid-19 pandemic curtailed normal teaching activities, allowed us to (re-)design an assignment to align problem based learning approaches. Specifically, students were given a serial problem and asked to submit a solution that ran faster, whether to parallelise using MPI (taught physically pre-covid-19) or OpenMP (which had been covered via remote teaching) or indeed to use compiler optimisation techniques (covered earlier in the course).

We noted there are a number of leading conferences covering various fields of high performance computing (HPC), notably SC [2] and ISC [3], that run a "Student Cluster Competition". At Liverpool, it has been identified that these competitions offer learning opportunities for students. Specifically, by participating, students would practise their technical skills but also soft skills such as group work, meeting deadlines and report writing. However, there are specific barriers to directly using the competitions for students learning HPC, including: the fixed time of international conference competitions; rules limiting the number of teams per institution; potential privacy issues relating to submissions and marking of assignments; and potential conflict of IP ownership.

Defining the Assignment

We therefore have designed assignments that take the principles of student cluster competitions, but are designed to fit the constraints of university teaching. In outline, students are assigned to a given small group, and all groups are given a specific open source software code with default build

instructions, that they need to improve. They receive two sets of data, a "debug" dataset and a "production" dataset, and:

- groups have a fixed deadline and it is up to each group how much effort to expend on the project
- codes can be run on the HPC architectues provided; these are specifically named (to avoid confusion) and include the university HPC resources (100 nodes of Intel Skylake, 1 node with 2 Nvidia V100 cards, 4 nodes of Intel Knights Landing), augmented by the department's HPC dev resources (1 node with 2 Nvidia Quadro cards, 1 node with Xilinx U200 FPGA card). Leveraging our collabations with MicroSoft, future runs of the assignments will provide given MS Azure IaaS plaforms.
- numerical results for both datasets have to agree within roundoff error analysis compared to baseline output. This is defined as the output when the code is compiled with the stated default build instructions and run on the Skylake cluster.
- the optimised code has to submitted to the department's gitlab site, together with full build instructions (e.g. Makefile).

Unlike some conference competitions, we do not impose power caps, primarily due to the challenges in providing "energy to solution" data to students for every platform. The key aim for a student group is to obtain (correct) results as fast as possible. They also have to write a concise two page formal report outlining their submitted solution.

Assessing the Assignment

The main aim of the assignment is to support learning by students. The work is assessed which requires a clear marking scheme, viz:

- speed: 70% based upon raw speed of the modified code when running the production dataset, defined in terms of quickest time to solution within roundoff accuracies.
- approach: 10% available for novel approaches.
- report: 20% for clear, concise, explanation of the approach taken and discussion of the code improvements.

IV. Broadening Participation via Computer Systems Genome Research Group

By: Margaret Ellis, Kirk Cameron, Godmar Back

The mission of the Computer Systems Genome Project (CSGenome) is to conduct the first scientific effort to catalog the lineage of computer system performance over time to enable knowledge discovery and further understanding of the impact of computing innovations on transformative technologies, the knowledge-based economy, and societal change. This work is powered by a large group of undergraduate research students with the specific intention to engage a broader group of students in computer systems research.

Facilitation of this supportive research team has propagated the recruitment and retention of a diverse set of students in an area of computer science that traditionally lacks such representation of women, black students, and students with disabilities. This talk will address team culture, management, and communication strategies that enable an encouraging learning and research environment for a diverse group of undergraduate students. The supportive community atmosphere promotes a sense of belonging for students and acquisition of practical skills develops their self-efficacy. Several such students have chosen to pursue graduate studies in computer systems.

A. Significance of the Work

Since the dawn of computing, the world has tracked system performance. Yet, computer system performance data is still primarily siloed by benchmark, system, or system component. The CSGenome project (https://csgenome.org/about/ was born from lack of a central repository for systems component, configuration, and benchmarking data recognized by the VarSys Team https://varsys.cs.vt.edu/the-varsys-team/. We began with work to support that team and a goal to engage a broader group of students in computer systems research.

In its first few years our research team has consisted of 40% women, 10% black students, and 5% students with disabilities. This is significantly more diverse than our overall computer science department which is 19% women and 3% black students. Notably, this is also extremely more diverse than the computer systems research community. We began building the undergraduate team in the Spring of 2018 and four students are currently in progress towards CS graduate degrees in the systems area. Three students obtained competitive undergraduate internships and subsequent full-time positions. Alumni continue to engage with the group as valuable role models.

Our success in recruiting and retaining students is multifaceted. Students perceive the skills and experience acquired as very useful to their future careers and also find the long-term project goals interesting and relevant. The team is explicitly supportive, and not intimidating, so that even when the content may seem complicated or overwhelming students are encouraged to persevere and are provided with resources, feedback, and peer role models. Early CS research experience in an active research group with a focus on community building and support has been shown to yield positive outcomes on students' perception of and retention in Computer Science [6]. We provide community building and support specifically within a systems research group.

Key components of our team culture align with the 2018 report "Women in computer systems research: increasing community, awareness, and communication" [7]. Students have regular casual interaction with each other and faculty. Time and space (whether physical or virtual) are provided so that the team has open-ended working time together to allow for unstructured friendly conversations and incidental learning. Also instrumental is that in addition to larger group meetings and more formal presentations and code or design reviews, the students have ready access to faculty.

The team culture is that students are expected to collaborate and help each other, the more advanced students create tutorials with faculty consultation and new students improve material as they work through it. In our recent all-online work environment, we have pre-recorded video tutorials by graduate students for the team and we record informative zoom sessions. We consciously strive to maintain a transparent and open environment to build a sense of community for the students to learn and thrive. On a larger scale our project aims to influence the perception of and communication about computer systems research to a broad audience by making our repository, analysis, and educational materials available through the project website.

Some of the efforts that have made this approach a success include:

- Recruitment of a diverse set of students enrolled in CS2-level courses with options to volunteer, earn undergraduate research credit, or become a wage employee.
- Practical skills instruction and training to acclimate students to version control, Linux, development environments, python, and project subgroup-specific technologies.
- Team meeting and working sessions with various subgroups (e.g. front-end development, memory component data pipeline, new-to-CSG, outreach) for brainstorming, planning, problem solving and reporting.
- Team communication using discord, git issues, emails, and weekly updates via google docs with faculty feedback.
- Research skills training to introduce students to literature surveys, publication reviews, research methods, and LaTeX.
- Cultivation of peer-to-peer relationships by assigning tasks to pairs of students, providing in-group mentors for students, and communication with recent graduates.
- Faculty roles include project vision, technical expertise, advising, and facilitation.

V. A Master Degree Course in Computational Engineering at University of Warsaw

By: Piotr Bała

A. Introduction

Computer science studies are very popular in Poland and other countries. They are conducted at many universities both in the field of mathematical sciences (mainly universities) and technical sciences (primarily technical universities)[1]. In Poland, computer science students are the most numerous group divided into a field of study. The number of candidates has been oscillating at the level of several candidates for one place for many years, taking into account the relatively rational assessment of opportunities based on the results of secondary school leaving examinations by young people.

A large number of computer science students cannot keep up with the market demand which significantly exceeds the number of graduates. Besides, the demand for IT specialists is diverse and changes with the emergence of new technologies. Unfortunately, education in this area does not keep up with the demand, which is particularly visible in the field of technologies related to computer simulations, large-scale calculations or the processing of large data.

The University of Warsaw, founded in 1816, is the largest and leading university in Poland, with about 2,900 academic staff among its 5,300 employees, and over 50,000 students. Offering courses in 32 fields of arts and sciences, its 18 faculties include natural sciences, social sciences, humanities, and over 30 extra-departmental and inter-faculty centers and programs. The University of Warsaw is participating in numerous research projects.

ICM, The Interdisciplinary Centre for Mathematical and Computational Modelling, a basic unit within the University of Warsaw, founded in 1993, is a research center in computational sciences and a center of HPC infrastructure. The research at ICM, of the strongly interdisciplinary profile, encompasses computational and information sciences, with special focus on their mathematical foundations and applications in other areas of science, technology, and e-economy.

B. Computational Engineering

Computational engineering is a second level (according to the Bologna Process) master degree program launched in the winter semester 2016/2017. The profile of the program is practical, which means that at least 50% of the classes is provided in the form of laboratories, seminars, and other practical activities. This includes a 3-month internship performed in the industrial environment including HPC centers. The studies are addressed to graduates of engineering studies (first level, 7-semester studies, completed with an engineering degree) or master's degrees. The studies last 3 semesters and end with obtaining a master's degree. Initially, the studies were located in the field of technical sciences, from the 2019/20 academic year, the studies are conducted as practical studies in the discipline of computer science in the field of natural sciences.

The study program deals with the use of scientific simulations to solve advanced scientific and technical problems. It is the answer to the growing demand for the use of computer simulations in various fields of science, economy, and business. Computer modeling is necessary to build a knowledge-based economy. As already mentioned before, the demand for specialists in this field is reported by all large enterprises operating in the field of digital technologies.

A significant part of the classes are lectures selected by students according to their interests. The range of classes to choose from is diverse, although it largely depends on the lecturers available. Recently, selected on-line courses on

[1]In Poland there an official list of disciplines and research fields. Each university program has to belong to one or more fields and disciplines. For computer science, there is a distinction between theoretical and practical one. Each of them belongs to the different research fields .

machine learning and artificial intelligence emerged as part of the elective courses supported by final projects implemented at the University of Warsaw. To enrich the offer, elective classes are common to students from subsequent training cycles. Each semester, at least one of the lectures is conducted in English.

To date, we have not decided to conduct all classes in English, as to which there are no obstacles either on the side of the lecturers or on the side of the students. The studies, so far are directed primarily at Polish students, hence the use of Polish is natural. For now, interest in studying by foreign students is low and concerns people who know Polish. In the case of increased interest, it is possible to start studies entirely in English.

The study program has been entirely defined based on the learning outcomes. It is fully compliant with applicable regulations and industry trends.

Computational engineering studies are directed primarily at non-IT specialists, however, the experience of the first years showed that mathematical and computer science background is very important. Lack of such preparation constitutes a significant barrier hindering passing mathematical subjects. Therefore, during recruitment, priority is given to graduates of fields of study run by university faculties who have the right to grant doctoral degrees and provide basic IT knowledge. The list of faculties includes computer science; applied computer science; electronics and telecommunication, automation and robotics; biomedical engineering, mechatronics, electrotechnics, physics, applied physics, and others connecting these areas of knowledge.

Computational engineering studies are full-time studies, therefore they are free for Polish citizens and EU countries. The exemption from fees also applies to persons from other countries mentioned above who have a Polish Card - they are people of Polish origin, who often speak Polish but live in Belarus, Ukraine, Kazakhstan or other non EU countries. For the remaining students, the studies are paid, the tuition fee is 1,500 Euro per semester (4,500 Euro for the entire studies).

The computational engineering curriculum consists of compulsory and optional lectures, including lectures in the humanities or social fields. For example, in the first semester, students implement two compulsory subjects: modern computing, database and network systems (9 ECTS points[2]) and parallel computimg (6 ECTS).

The lectures include an introduction to high power computing systems, presentation of basic computer architectures, processors, networks, and data storage systems. Students will learn about queuing systems and their use in practice, learn about profiling technologies and code optimization both scalar and vector. They will learn about the principles of compiling and installing software or numerical libraries. The lecture includes an introduction to grid and cloud technologies, virtualization, and computing portals. Dedicated classes are on Hadoop and Spark technology. During one of the first classes, students learn practically how to use HPC systems available at ICM University of Warsaw. These classes, from the 2019/2020 academic year, are separated in the schedule (10 hours, 1 ECTS point). As part of the lecture, one-day workshop is carried out, dedicated to installing selected applications on ICM computing systems. These classes are a practical verification of students' knowledge and skills. In a result, students prepare reports from the installation process including its verification and performance measurements.

In the first semester, there are also classes in parallel programming. In the era of multiprocessor computers, this subject is important in educating users of high-power computers. The course program includes a theoretical introduction to parallel programming (8 hours) and practical classes divided into three blocks: programming in the PGAS model using the PCJ (Parallel Computing in Java) library (12 hours), programming in the messaging model (MPI, 20 hours) and programming in a model with shared memory (OpenMP, 20 hours).

The second semester of study includes two compulsory lectures on computer simulations in natural sciences (26 lecture hours, 4 hours of classes, 3 ECTS), and the basics of modeling in social sciences (26 hours of lectures, 4 hours of classes, 3 ECTS). Both lectures were intended as a review of the applications of computer simulations in the most important fields of science.

The presented study results show that the computational engineering program equips students with basic domain knowledge plus the necessary skills and in-format competences in the field of large IT infrastructures. What's more, the education obtained in this way allows one to effectively solve complex problems that require the processing of large data or computer simulations.

Last but not least, one of the goals of program was to educate people for employment in supercomputing centers, including ICM. Despite the small number of graduates. It can be said that this goal has been achieved.

[2]ECTS - European Credit Transfer System – (the so-called credit points) – it is a system of clear and comparable marks and academic degrees which makes it possible to recognize diplomas in the European job market.

REFERENCES

[1] liverpool.ac.uk/centre-for-innovation-in-education/curriculum-resources/overview/ [14/09/20]

[2] sc20.supercomputing.org/program/studentssc/ [14/09/20]

[3] hpcadvisorycouncil.com/events/student-cluster-competition/ [14/09/20]

[4] M. Ignatova, The Top 10 Skills You Will Be Hiring for in 2017. *LinkedIn* 25.11.2016 https://business.linkedin.com/talent-solutions/blog/trends-and-research/2016/the-top-10-skills-you-will-be-hiring-for-in-2017 Accessed: 20.05.2019.

[5] *Hackathon Great Programming Challanges* http://wwp.icm.edu.pl [07/06/20]

[6] M. Barrow and S. Thomas, and C. Alvarado, A Structured CS Research Program for Early-College Students. in Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'16). Association for Computing Machinery, New York, NY, USA, pp 148 – 153.

[7] L. Golubchik and J. Weston, Women in computer systems research: increasing community, awareness, and communitation, in Technical Report. National Science Foundation, USA, 2018, pp. 1-21.