Addendum No. 4

ACADEMIC VACANCIES PUBLIC INFORMATION

Field Name	Details
Faculty	Mathematics and Computer Science
Department	Computer Science
Position number in the establishment plan	100
Position	Teaching assistant (3-year fixed-term contract)
Academic subjects in the position description/ research areas, as listed in the establishment plan	Algorithms and Programming (English) Fundamentals of Programming (English) Object Oriented Programming (English) Data structures and algorithms (English)
Branch of science	Computer science
Position opening description	Teaching assistant, 100, Department of Computer Science. The position of teaching assistant involves carrying out teaching activities, scientific research and student coaching, as well as providing services to the academic community. Applicants for the vacant position of teaching assistant must have a scientific expertise in line with the standards for the Computer science field and the disciplines of the position. Applicants must also include proof of proficiency in English (C1 level or documents attesting to studies or research placement abroad for a total duration of at least 9 months, in education or research institutions where English was the main language used).
Responsibilities	Teaching activity: seminar, laboratory, projects, coaching, assessments, exams, creating teaching materials for the disciplines of the position. Scientific research activity: • participation in at least one research seminar within the faculty; • participation in research grants competitions; • publication, in each 4-year period of at least 5 BDI indexed articles/studies (Mathematical Reviews/ MathSciNet, ZMath (Emis), Computing Reviews, IEEE Xplore, DOAJ, SCOPUS, DBLP) of which at least 2 are ISI or SCOPUS indexed or published in relevant international conference proceedings (ACM, IEEE, AMS, EMS). The student coaching activity: supervising theses, tutoring groups of students, coordinating students' participation in the activity of scientific circles and in student competitions. Services for the academic community: taking part in actions carried out by the department, faculty and university (promotion of the admission, collaboration with the economic environment, etc.).

Date and time of the oral examination	28.01.2020 at 12:00 p.m.
Oral examination location	Department of Computer Science 58-60 Teodor Mihali Street, room C335
Examination date, time and place for academic position opening	Selection process for the TEACHING ASSISTANT academic position: 1. Assessment of application file; 2. Oral examination; 3. Written examination. The oral examination consists in delivering a seminar/ laboratory/ practical work project presentation. Based on the position theme and bibliography, the Committee establishes the topic of the presentation to be delivered during the oral examination and communicates it to the candidates 48 hours prior to the examination date by email and by posting it on the faculty display board and website, indicating the date and time of posting, signed by the chair of examination committee. The minimum duration of the oral examination delivered by the candidate is 30 minutes; the exam must also include a question and answer session by the committee and/or the public; Exam 1 - Written test (in English): 28.01.2020, at 8:00 a.m., Department of Computer Science 58-60 Teodor Mihali Street, room C335. Exam 2 - Oral examination (in Romanian): delivering a seminar/ laboratory/ practical work project presentation - 28.01.2020, at 12:00 a.m., Department of Computer Science 58-60 Teodor Mihali Street, room C335. If there are several candidates, the committee will decide the order in which they will take the oral examination. The application file, oral examination and the written examination will be calculated in equal proportions towards the final score given in the application assessment report written by each member of the examination committee;
The topic and bibliography for the academic position selection process	Exam 1 – Written examination on a given topic: Fundamentals of programming, Object-oriented programming, Data structures 1. Fundamentals of programming Sub-algorithms: specification, testing. Algorithm classes: searching, sorting, interclass correlation. Algorithm design methods: top-down, successive refinement. Sub-programs, call and pass parameters (by value and reference). Programming techniques: Backtracking, Divide et impera, Greedy. Modular programming: module, interface, implementation; in C/C++, Java, Python. 2. Object-oriented programming

	Classes, objects. Inheritance, polymorphism. Interface based programming. 3. Data structures Abstract data type (ADT). ADT Array, Collection, List, Stack, Queue, Dictionary. Specification of ADT. Implementations for ADT using: vectors, linked lists, binary trees. Bibliography: 1. M. Frenţiu, B. Pârv, Elaborarea programelor. Metode şi tehnici moderne, ProMedia, Cluj-Napoca, 1994 2. M. Frenţiu, H.F. Pop, G. Şerban, Programming fundamentals, Cluj University Press, 2006 3. T. Cormen, C. Leiserson, R. Rivest: Introducere în algoritmi. Cluj-Napoca: Editura Computer Libris Agora, 2000 4. B. Eckel, Thinking in C++, vol I şi II, http://www.mindview.net 5. B. Eckel, Thinking in Java, http://www.mindview.net 6. M.A. Ellis, B. Stroustrup, The annotated C++ reference manual, Addison-Wesley, 1994 7. The Python language reference. http://docs.python.org/py3k/reference/index.html 8. R.S. Pressman, Software engineering. A practitioner's approach, 6th ed., McGraw-Hill, 2005 Exam 2 – Oral examination: delivering a seminar/ laboratory/ practical work project presentation Topics: A. Algorithms and programming 1. Introduction to software development processes 2. Procedural programming 3. Modular programming 4. User defined types 5. Principles of design and programming 6. Object-oriented programming 7. Software design 8. Software testing and inspection 9. Recursion 10. Algorithm complexity 11. Searching and sorting algorithms 12. Problem Solving Methods (I) - Backtracking, Greedy 13. Problem Solving Methods (II) - Divide & Conquer, Dynamic Programming Bibliography: 1. M.L. Hetland, Beginning Python: From Novice to Professional, Apress, 2005. 2. M. Frentiu, H.F. Pop, Fundamentals of Programming, Cluj University Press, 2006.

	3. K. Beck, Test Driven Development: By Example. Addison- Wesley Longman, 2002. http://en.wikipedia.org/wiki/Test-driven_development 4. M. Fowler, Refactoring. Improving the Design of Existing Code, Addison-Wesley, 1999. http://refactoring.com/catalog/index.html 5. The Python Programming Language - https://www.python.org/ 6. The Python Standard Library - https://docs.python.org/3/library/index.html 7. The Python Tutorial - https://docs.python.org/3/tutorial/ B. Fundamentals of programming 1. Introduction to software development processes 2. Procedural programming 3. Modular programming 4. User defined types 5. Principles of design and programming 6. Object-oriented programming 7. Software design 8. Software testing and inspection 9. Recursion 10. Algorithm complexity 11. Searching and sorting algorithms 12. Problem Solving Methods (I) - Backtracking, Greedy 13. Problem Solving Methods (II) - Divide & Conquer, Dynamic Programming Bibliography: 1. Kent Beck - Test Driven Development: By Example. Addison- Wesley Longman, 2002. 2. Kleinberg and Tardos – Algorithm Design. Pearson Educational, 2014 (http://www.cs.princeton.edu/~wayne/kleinberg-tardos/) 3. Martin Fowler - Refactoring. Improving the Design of Existing Code. Addison-Wesley, 1999. (http://refactoring.com/catalog/index.html) 4. Frentiu, M., H.F. Pop, Serban G. - Programming Fundamentals, Cluj University Press, 2006 5. The Python language reference. (https://docs.python.org/3/reference/index.html) 6. The Python standard library. (https://docs.python.org/3/library/index.html) 7. The Python tutorial. (https://docs.python.org/3/tutorial/index.html) C. Object-oriented programming 1. Basic elements of C language 2. Modular programming in C/C++ 3. Object oriented programming in C++. 4. Derived data types and user-defined types, dynamic allocation in C++. Elements of generic programming 5. Inheritance 6. Polymorphism

	8. Class hierarchies 9. Graphical user interface (GUI) 10. Event-driven programming (Events: Qt signals and slots; GUI design software; Callback/Observer) 11. Event-driven programming (Custom graphic components; MVC template; Predefined models) 12. Design templates Bibliography: 1. B. Stroustrup. The C++ Programming Language, Addison Wesley, 1998. 2. Bruce Eckel. Thinking in C++, Prentice Hall, 1995. 3. A. Alexandrescu. Programarea moderna in C++: Programare generica si modele de proiectare aplicate, Editura Teora, 2002. 4. S. Meyers. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition), Addison-Wesley, 2005. 5. S. Meyers. More effective C++: 35 New Ways to Improve Your Programs and Designs, Addison-Wesley, 1995. 6. B. Stroustrup. A Tour of C++, Addison Wesley, 2013. 7. C++ reference (http://en.cppreference.com/w/). 8. Qt Documentation (http://doc.qt.io/qt-5/). 9. E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Longman Publishing, 1995. D. Data structures and algorithms 1. Introduction Data structures. Static, semi-static and dynamic data structure. 2. Data types: domain, operations and data representation 3. Fields 4. Abstract Data Types - ADT Collection, ADT Dictionary, ADT Stack and Queue, ADT Priority Queue 5. Linked Lists - ADT List, Linked List 6. Heap 7. Hash Table 8. ADT Tree - Binary tree 9. Binary search tree 10. Self-balancing binary search trees 11. Applications and data structures in various programming languages (Python, C++, Java, C#) Bibliography: 1. T. Cormen, C. Leiserson, R. Rivest, C. Stein: Introduction to algorithms, Third Edition, The MIT Press, 2009 2. S. Skiena: The algorithms design manual, Second Edition, Springer, 2008 3. N. Karumanchi: Data structures and algorithms made easy, CareerMonk Publications, 2016 4. M. A. Weiss: Data structures and algorithm analysis in Java, Third Edition, Pearson, 2012 5. R. Sedgewick: Algorithms, Addison-Wesley Publishing, 1984
Description of the selection process	Examination Committee assesses candidates taking into the account the following criteria:

	• Content of the application file; • Oral examination (Exam 1) • Oral examination (Exam 2). The final grade of each applicant is calculated as the arithmetic mean of the scores obtained according to the above criteria. Each member of the committee (including the chair) draws up an individual assessment report giving a final grade for each applicant. Eligible applicants must obtain: • at least grade 6 (six) for each criterion; • the final grade at least 7 (seven) given by each reviewer; • average score at least 8.50 (eight and 50%). The chair of the selection committee draws up a report on the selection process in which they indicate the final grades assigned to the applicants by the members of the committee and the average score obtained by each applicant, calculated as the arithmetic mean of the final grades assigned in the individual assessment reports. The average score thus obtained constitutes the result of the selection competition for each applicant. The selection competition committee establishes the ranking of applicants based on the final average score and nominates the eligible applicant who has obtained the best result in the selection competition. The chair of the selection competition committee submits the report on the selection competition to the secret ballot of the members of the committee. Following the secret ballot, the chair takes note of the result of the vote, communicates it to the members of the committee and indicates it at the end of the report on the selection competition, specifying the number of votes "for" and "against", respectively, the votes cast remaining secret. If the "for" vote is not given by a majority of the members of the committee, the position put up for the selection competition will not be filled by any of the candidates. The report on the selection competition is signed by each member of the selection competition committee and by the chair of the committee:

Head of department,

Professor Laura DIOȘAN, PhD