Программа учебной дисциплины «Алгоритмы и структуры данных 1»

Утверждена
Академическим советом ООП
Протокол № от «____»__________ 20____ г.

Авторы	Р. Э. Яворский, С. А. Шершаков
Число кредитов	9
Контактная работа (час.)	144
Самостоятельная работа (час.)	198
Курс	1
Формат изучения дисциплины	без использования онлайн-курса

1 Course Description

1.1 Title of a Course

Algorithms and Data Structures.

1.2 Pre-requisites

1.2.1 Part I

As a matter of fact, good English and Math are usually enough to enter the program, so no preliminary knowledge in data science or specific programming skills are required, because some of the students may have zero experience in these.

1.2.2 Part II

Successful completion of the first part of the course is the sole prerequisite for being enrolled for the second part.

1.3 Course Type

Compulsory.

1.4 Abstract

The training course "Algorithms and Data Structures" is offered to students of Bachelor Program "HSE and University of London Double Degree Programme in Data Science and Business Analytics" (area code *01.03.02*) at the Faculty of Computer Science of the National Research University — Higher School of Economics (HSE). The course is classified as an *compulsory subject* (*Б.Пр.Б* unit / base module, *Б.Пр* – Major disciplines of 2018–2019 academic year working curriculum); it is a two-module course (semester A quartile 2 and semester B quartile 4).

The course is divided into two logical parts, which do not basically depend on each other. *The first part* is given during semester A quartile 2 under responsibility of Dr. Rostislav Yavorskiy. *The second part* is given during semester B quartile 4 under responsibility of Lect. Sergey Shershakov.

The syllabus is prepared for teachers responsible for the course (closely related disciplines), teaching assistants, students enrolled in the course as well as experts and statutory bodies carrying out assigned or regular accreditations.

1.4.1 Part I

The first part of the course is intended to be taught during the second module (quartile) of the program, so it could be treated as introduction into the subject of "Applied data analysis" for students who just entered the program.

The lectures and seminars are mostly independent. Lectures are focused on general topics of software engineering, data science, software development project management, etc. The seminars are targeted at teaching basics of Python programming language. Much attention is given to development of learning skills, that is why the grading formula encourages different kind of self-study, collaboration and team work.

1.4.2 Part II

The *second part* of the course is intended to be taught during the fourth module (quartile) of the program. It is dedicated to the basics of design and analysis of algorithms. The part involves learning fundamental data structures implemented by the C++ Standard Library (STL).

The lectures and practical classes are closely inter-related. The lectures are primarily intended to introduce new topics, whereas the practical classes are intended for solving specific problems by coding programs in C++.

2 Learning Objectives

During the course "Algorithms and Data Structures" the participants will:

- study algorithm design princiles;
- learn how to analyze the complexity of an algorithm in terms of time and space;
- study design principles of basic data structures;
- practice implementing algoritms and data structures as C++-applications.

3 Learning Outcomes

Students who complete this course successfully will learn or acquire:

(Technical skills)

- basic concepts and methods of designing algorithms;
- skills is designing data structures by using C++;
- approaches to analysis of algorithms complexity in terms of time and space.

(Soft skills)

- improve team-working skills with using collaborative working tools;
- improve presentation skills;
- improve skills on writing reports and technical documentation, including rapidly changing documentation with using wiki and other specific tools;
- improve self- and peer-review skills.

4 Course Plan

4.1 Part I: semester A quartile 2

#	Topic Name
1	Euclidean algorithm for the GCD; Proof of correctness. Solution of quadratic equation. Hilbert's tenth problem (1900). Turing machine (1936). Church–Turing thesis. Time complexity of algorithm. Space complexity of algorithm.
2	Time complexity of an algorithm. Big-O notation. Array data structure. Comparison of list data structures (Linked list, Array, Dynamic Array, Balanced Tree, Random Access List, Hashed array tree).
3	Lists vs Sets. Bubble chart. A set of 4-vectors. Graph data structure. Social and professional networks. Networks of organizations.
4	Graph visualization and analysis.
5	Bubble chart. Graph connectivity.
6	Metric space. Hamming distance. Bipartite graph (bigraph).
7	Tree. Binary tree. Concrete syntax tree, parsing. Abstract syntax tree. Hierarchical clustering. Decision tree.
8	Search in a graph/tree. Breadth First Search. Depth First Search.
9	Blockchain architecture. Key features of blockchain database. Distributed computing. Mining.

Notes:

1. Each sequential number above corresponds to a separate theme, whereas a theme can span over one or more lectures and/or practical classes.

4.2 Part II: semester B quartile 4

#	Topic Name
1	MergeSort, Recurrences, and Asymptotics.
2	Solving recurrences and the master theorem.
3	Recurrence relations (substitution method, recursion trees). The Selection problem.
4	Heaps.
5	Quicksort, Probability and Randomized Algorithms.
6	Sorting Lower Bounds, BucketSort, RadixSort.
7	Red-Black Trees.
8	Hashing.
9	More on Graphs. Topological Sort.
10	Strongly connected components, Dijkstra's algorithm.
11	Dynamic Programming and shortest paths: Bellman-Ford and Floyd-Warshall.
12	Examples of dynamic programming: Longest common subsequence, Knapsack, Independent Set.
13	Greedy algorithms, activity selection problem.
14	Minimum spanning tree: Boruvka, Kruskal, Prim.
15	Minimum Cut: Karger's Algorithm.
16	Max Flow and the Ford-Fulkerson Algorithm.

Notes:

1. Each sequential number above corresponds to a separate theme, whereas a theme can span over one or more lectures and/or practical classes.

5 Reading List

Required

1. Thomas H. Cormen et al. *Introduction to Algorithms.* 3rd ed. The MIT Press, 2009.

6 Grading System

The course grade is based on both ongoing assessment and final examination. Every module ends up with an final exam. The grade for the exam together with a cumulative grade represent a final grade for the module. The ultimate grade G for the whole course is calculated as:

$$G = 0.7 \cdot \min(P_1, P_2) + 0.3 \cdot \max(P_1, P_2), \tag{1}$$

where P_1 is a *first part* final grade, and P_2 is a *second part* final grade.

Grade G is rounded (up or down) to an integer number of points before entering them into records. P_1 and P_2 are also rounded in (1). The conversion of rounded 10-point scaled results to 5-point scaled ones is performed according to Table 1.

Table 1: Correspondence of ten-point to five-point marks

Ten-point scale [10]	**Five-point scale [5]**
1 — unsatisfactory 2 — very bad 3 — bad	Unsatisfactory — 2
4 — satisfactory 5 — quite satisfactory	Satisfactory — 3
6 — good 7 — very good	Good — 4
8 — nearly excellent 9 — excellent 10 — brilliant	Excellent — 5

6.1 Part I Grading Details

The final grade of the *first part* is computed by the following formula:

$$P_1 = 0.2 \cdot E_1 + 0.4 \cdot CS + 0.4 \cdot AS = 0.2 \cdot E_1 + 0.8 \cdot (0.5 \cdot CS + 0.5 \cdot AS), \tag{2}$$

where E_1 is a grade for the *first part exam*, which takes place at the end of the quartile 1 (semester A), CS is a score for *coding skills* and AS is a score for *analytical skills*.

Due to HSE rules the cumulative score P_1 must be a whole number.

The *first part final exam* will consist of several coding tasks and open-ended questions.

Coding skills are assessed during seminars of the first part (50 %) and the intermediate tests of the first part (another 50 %).

The maximal score for the analytical skills is 10, which could be accumulated from the following tasks:

- weekly blog posts in the online community (0.5 score points for each),
- streaming video to present solution of a coding task in Python (2 points),
- online video presentation of a Python library (3 points),
- team project on development of online analytical service with Django or Flask (5 points).

6.2 Part II Grading Details

The *final grade* P_2 for the *second part* is calculated as follows:

$$P_2 = 0.4 \cdot E_2 + 0.6 \cdot OA, \tag{3}$$

where E_2 is a grade of the *second part exam*, which takes place at the end of the quartile 3 (semester B), *OA* is an *ongoing assessment* grade (both 10-point scale). The ongoing assessment *OA* measures participant's performance throughout all classes and involves various types of activities (see Sect. 6.3).

The *final exam for the second part* as well as the intermediate tests are given in the form of a written test (paper- *or* computer-based, *subject to further clarification*). One (10-point scale) grade is given for the exam.

6.3 Ongoing Assessment

The ongoing assessment grade is accumulated throughout all the classes and is related to a participant's activity. An ongoing control structure is individual for every class.

During the classes, there are some activities available for students to be involved in. They include (but are not limited by) writing code and developing applications, evaluating practical problems, solving tests, answering questions and so on. Every activity is evaluated and grants some points (*RP*) to participants. We consider two sorts of points: 1) regular points (*RP*) and 2) bonus points (*BP*).

As a result of every class, a certain maximum number of *RP* can be earned (RP_{max}). *RP*s are accumulated during a module and give a resulting grade as maximum **10** (10-point scale) to *OA* only if every class brings a maximum number of points (RP_{max}) for a given participant. *BPs* are given for additional efforts and for excellent jobs. They are also accumulated and together with *RPs* give the following formula for calculating *OA* component of formula 3:

$$OA = 10 \cdot \frac{RP + BP}{RP_{max}}, \tag{4}$$

Thus, if $BP = 0$, then the *OA* is ranged from 0 to 10. If $BP > 0$, the ratio $(RP + BP)/RP_{max}$ can be greater than 1. In the latter case the value *OA* is limited by 10.

Finally, some kinds of out-of-class activities can be accounted for as a part of ongoing assessment. *Peer review work*, *preparing and reporting one of a course-related topics* are examples of such activities.

6.3.1 Regular tests

Students' skills in programming are tested using automated testing. This way, a student is assigned an individual task, prepares it by using a personal computer and, then, submits it by using a special service, such as Yandex.Contest or a repository-based tool. The specific solution is subject to further clarification.

The individual home-based task submissions are to be further reassessed through in-class tests or examinations.

For any two corresponding submissions, one for home work and one for class work, graded as H and C respectively, the resulting grade R is calculated as follows:

$$R = 0.8 \cdot \min(H, C) + 0.2 \cdot \max(H, C). \tag{5}$$

7 Guidelines for Knowledge Assessment

The knowledge gained by the students is systematically and consistently assessed throughout the course, which includes understanding and attendance rate of the lectures, as well as tests taken during practical classes. A quantitative basis is provided elsewhere. The instructors are supposed to check with the students highly probable coding mistakes and drawbacks in advance.

8 Methods of Instruction

Learning happens through traditional face-to-face lectures (classroom presentations being distributed through a repositiry) and consolidation of the delivered knowledge and getting hands-on experience at practical classes.

9 Special Equipment and Software Support (if required)

Students are highly reccommended to use their own laptops with pre-installed and configured software, if possible. The computers in computer classes are also suitable for performing programming tasks. The exact set of software needed for the courses will be listed in an associated educational service, such as LMS or wiki.

References

[1] Thomas H. Cormen et al. *Introduction to Algorithms*. 3rd ed. The MIT Press, 2009.

Author of the program: ____________________ Rostislav Yavorskiy

Author of the program: ____________________ Sergey Shershakov