Программа учебной дисциплины «Алгоритмы и структуры данных 1»

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

Автор	С. А. Шершаков
Число кредитов	9
Контактная работа (час.)	166
Самостоятельная работа (час.)	166
Курс	1
Формат изучения дисциплины	без использования онлайн-курса

1 Course Description

1.1 Pre-requisites

Successful completion of "Introduction to Programming" course is the sole prerequisite for being enrolled for this course.

1.2 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 2019–2020 academic year working curriculum); it is a two-module course (semester B quartiles 3 and 4).

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.

The course is dedicated to the basics of design and analysis of algorithms. It also 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 principles;
- 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

#	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	Tree. Binary tree. 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. The third module ends up with an intermediate test. The fourth module ends up with a final exam. The grade for the final exam together with a cumulative grade represent a final grade for the course. The ultimate grade G for the whole course is calculated as:

$$G = 0.4 \cdot E + 0.6 \cdot OA, \tag{1}$$

where E is a grade for the final exam and *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. 6.1).

Grade G is rounded (up or down) to the nearest integer number of points before entering them into records.

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]**
0 — nothing to assess 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 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*). *BPs* are given for additional efforts and for excellent jobs.

*RP*s and *BP*s are accumulated during a semester. At the end of the semester *OA* is calculated according to the following formula for calculating *OA* of formula (1):

$$OA = \max\left(\left\lceil 10 \cdot \frac{RP + BP}{RP_{max}} \right\rceil, 10\right), \tag{2}$$

where, RP_{max} denotes the maximum possible number of points that can be taken during the module.

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.

6.1.1 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:

$$R = 0.8 \cdot \min(H, C) + 0.2 \cdot \max(H, C). \tag{3}$$

6.2 Other conditions

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

The second re-sit of each of the final exams is provided by a board of professors. There is allowed to omit any ongoing assessment when calculating the ultimate grade.

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: ____________________ Sergey Shershakov