Приложение 3
УТВЕРЖДЕНО
приказом МФТИ
от 11.12.2019 г. № 2094-1

Appendix 3
APPROVED
by MIPT order № 2094-1
December 11, 2019

Entrance Examination in Computer Science
for students to be readmitted after expulsion or be transferred from other universities to MIPT
(major 01.03.02 “Applied mathematics and Computer Science”)

Test Format

The entrance test shall take the form of a combination of a practical and oral examination.

1. The exam takes place in accordance with the schedule established by MIPT order.

2. Only students having a valid Exam Sheet issued by the Admissions Office are allowed to sit a test.

3. To complete the exam practical part an ICT room with installed software is used: a text editor or development environment, a C++ compiler and a Web browser with access to the resource *contest.yandex.ru*

4. The total practical test time is 2 hours, with no breaks between them.

5. In the practical part of the entrance exam 5 tasks are required to be solved by test takers to be readmitted or be transferred to semester 2 and 6 tasks by test takers to be readmitted or be transferred to semester 3 through semester 8. All the tasks are checked by Automated Testing System.

6. The oral part of the entrance exam include
- the discussion of the tasks which were sent to Automated Testing System but were unsolved by test takers;
- an answer to an exam card containing theoretical part (the total preparation time is 1 hour).

7. The exam cards in theory consist of
- 2 questions from “Programming and algorithms” section when readmitting or transferring to semester 2 through semester 3;
- 2 questions: one from “Programming and algorithms” section and one from “Formal languages and translations” when readmitting or transferring to semester 4 through semester 6;
- 3 questions from various sections when readmitting or transferring to semester 7 through semester 8.

8. Test results are reported as a band score on a ten-point scale, with 0-2 being unsatisfactory, 3-4 being satisfactory, 5-7 being good, and with 8-10 being excellent.

Entrance Examination in Computer Science

"Programming and algorithms". Theoretical questions.

semester 2

1. Algorithm. Model of computation (example: single-tape Turing machines).
2. Algorithm. Algorithm complexity (time, space complexity). Recursion.
3. Sorting: problem statement. Sorting algorithms: bubble sort, insertion sort, Shell sort. Average-case and worst-case complexity.
4. Heapsort. Average-case and worst-case complexity.
5. Merge sort. Average-case and worst-case complexity.
6. Radix sort. Average-case and worst-case complexity.
7. Data structures. Data structures operations and their complexity. Abstract data type.
8. Array, list, singly linked list, circular linked list, stack.
9. Queue. Priority queue.
10. Heap.
11. Object-oriented programming. The C++ language.
12. Dynamic programming method. An example of a problem that is solved by dynamic programming. Analysis of complexity.

semester 3

1. Algorithm. Algorithm complexity (time, space complexity). Recursion.
2. Sorting: problem statement. Sorting algorithms: bubble sort, insertion sort. Heapsort. Merge sort. Quick sort. Radix sort. Average-case and worst-case complexity.
3. Data structures. Data structures operations and their complexity. Abstract data type.
4. Array, list, singly linked list, circular linked list. Stack, queue, deque.
5. A binary heap. Priority queue.
6. Search tree. Cartesian tree. AVL tree. Red-black tree. Tree with an implicit key.
7. Hash table. Collision resolution: separate chaining method. Collision resolution: open addressing.
8. RMQ. Sparse table. Segment tree.
9. LCA. Binary lifting approach. Reduction from RMQ to LCA and vice versa.
10. Graph. Directed graph. Graph representations. Graph traversal algorithms: depth-first search and breadth-first search. Topological sort. Counting the number of paths in a directed graph.
11. Strongly connected components. Kosaraju's algorithm. Tarjan's algorithm.
12. Finding the shortest paths in a graph. Dijkstra's algorithm. Bellman-Ford algorithm. Floyd's algorithm. A* algorithm. Heuristics.
13. Minimum spanning tree. Prim's algorithm.
14. Disjoint set union. Kruskal's algorithm.
15. Boruvka's algorithm.
16. Flow, Ford-Fulkerson algorithm. Edmonds-Karp algorithm. Dinic's algorithm.
17. Basic concepts of OOP. Constructors/destructors. Method overloading. Method hiding. What methods and operators are necessary to use a type as a parameter of a standard template container? "virtual" and "const" keywords. What is object slicing? Multiple inheritance.
18. C++ exceptions. Throw and catch exceptions. Error handling in constructors and destructors.
19. C++ templates.
20. STL containers under the hood, basic operations and their cost, usage specifics: vector, list, deque, stack, map, set, bitset and vector, unordered_map, priority_queue.
21. STL: iterators. What is an iterator? Iterator categories. What is a random access iterator?
22. Sorting and searching in STL. Which containers store items following a specific order? Heap in STL. Associative array. Interface, variants of implementation - hash-tables, rb-tree.

semester 4 through semester 8

1. Algorithm. Model of computation (example: single-tape Turing machines).
2. Algorithm. Algorithm complexity (time, space complexity). Recursion.

3. Sorting: problem statement. Sorting algorithms: bubble sort, insertion sort. Heapsort. Radix sort. Merge sort. Average-case and worst-case complexity.
4. Data structures. Data structures operations and their complexity. Abstract data type.
5. Array, list, singly linked list, circular linked list, stack. Queue. Priority queue.
6. Object-oriented programming. The C++ language.
7. Graph. Directed graph. Graph representations. Graph traversal algorithms: depth-first search and breadth-first search. Topological sort. Counting the number of paths in a directed graph.
8. Strongly connected components. Tarjan's algorithm.
9. Finding the shortest paths in a graph. Floyd's algorithm. Dijkstra's algorithm. Bellman-Ford algorithm. A* algorithm. Heuristics.
10. Minimum spanning tree. Prim's algorithm. Binomial heap. Amortized cost. Fibonacci heap.
11. Disjoint set union. Kruskal's algorithm.
12. Flow, Ford-Fulkerson algorithm.
13. Search tree. Cartesian tree. Fenwick tree. Sparse table and segment tree for RMQ. Reduction from RMQ to LCA and vice versa. Search for multiple minimums on a segment.
14. Substring searching. Rabin-Karp algorithm. Finite-state machine. Boyer-Moore algorithm. Knuth-Morris-Pratt algorithm.
15. Aho-Corasick algorithm. Suffix tree. Trie. Ukkonen's algorithm. Suffix array.
16. Basic concepts of OOP. Constructors/destructors. Method overloading. Method hiding. What methods and operators are necessary to use a type as a parameter of a standard template container? “virtual” and “const” keywords. What is object slicing? Multiple inheritance.
17. C++ exceptions. Throw and catch exceptions. Throw-lists. Editing throw-lists in overridden methods. Error handling in constructors and destructors.
18. C++ templates. STL sequence containers and adapters. What is an adapter over an STL container?
19. STL: iterators. What is an iterator? Iterator categories. What is a random access iterator?
20. STL containers under the hood, basic operations and their cost, usage specifics: vector, list, deque, stack, map, set, bitset and vector, unordered_map, priority_queue.
21. Sorting and searching in STL. Which containers store items following a specific order? Heap in STL. Associative array. Interface, variants of implementation - hash-tables, rb-tree. Implementation in the language.

“Formal languages and translations”. Theoretical questions.

semester 4 through semester 8

1. Nondeterministic finite state machines. Different variants of the definition. Deterministic finite state machines. Their equivalence.
2. Regular expressions. Kleene's theorem on the equivalence of regular expressions and finite automata.
3. Minimization of finite automata. Minimization algorithm. An algorithm for checking the equivalence of regular expressions.
4. Generative grammars. Chomsky hierarchy. Linear, context-free, context-sensitive grammars (definitions). Equivalence of linear grammars and finite automata.
5. Context-free grammars. Chomsky normal form for context-free grammars.
6. Pushdown automata. Different variants of the definition. Equivalence of pushdown automata and context-free grammars.
7. Pumping lemmas for regular and context-free languages. Examples of languages that do not lie in these classes.
8. Parsing algorithms for context-free grammars. Kock-Younger-Kasami algorithm and Earley parser.

“Machine learning”. Theoretical questions.

semester 7 through semester 8

1. Basic concepts of machine learning. Standard problems (classification, regression, clustering). Examples of quality metrics. Examples of simple algorithms solving standard problems: kNN, K-Means, naive Bayesian classifier.
2. Quality metrics in classification and regression problems (accuracy, precision, recall, F-measure, ROC-AUC, logloss, MSE, MAE, quantile loss, MAPE, SMAPE). Feature engineering: feature extraction, categorical feature encoding.
3. Linear methods of classification and regression. Loss functions and regularizers. Stochastic gradient descent method. Logistic regression optimization problem and estimation of class membership probability.
4. Linear methods of classification and regression. Support Vector Machine optimization problem.
5. Decision trees in the classification problem and in the regression problem. Decision tree ensemble: random forest and gradient boosting above the trees.
6. Decision trees in the classification problem and in the regression problem. Bias-variation trade-off (without proof). Analysis of boosting and bagging using bias-variation trade-off.
7. Neural networks, training (backprop), convolutional networks layers (dance, conv, pooling, batchnorm, dropout), nonlinearity (relu vs sigmoid, softmax), loss functions (logloss, l2, hinge).
8. Recurrent neural networks, training (backprop tt), the difference between recurrent and convolutional networks, recurrent layers (RNN, LSTM, GRU), examples of usage.
9. Problem of clustering. Agglomerative and statistical clustering methods. Lance-Williams formula, K-Means algorithm.
10. The problem of dimensionality reduction (reducing the dimension of the feature space). Principal component analysis (PCA) and tSNE (for both methods: basic idea, without proof).

Head of the Department of Algorithms and Programming Technology: **Victor V. Yakovlev**