PTI Final exam theses

2025 June

You need to pull a simple task next to the thesis, for which you have to write a program in a learned programming language (on paper). See tasks below.

All exams consist of 3 parts:

1. Thesis defense
2. The „easy task”
3. The pulled thesis

When defending your thesis, you must be able to answer the question(s) asked by the examiner, which will be available to everyone one week before the exam date (you can contact the institute secretary by e-mail for it). If it’s found, that the thesis was plagiarized, or was not written by the candidate, then the thesis defense grade is insufficient (the final exam’s as well).

During the exam the “easy task” has to be solved perfectly, aside from one or two syntax errors that occur due to programming on paper, - i.e. semantically perfect tasks with small syntactic errors are still acceptable – however the final exam grade is insufficient. The exercise has to be solved using any one of the following programming languages: C, Java. From these the candidate can choose freely. It is forbidden to use a function/method, which directly gives the solution to the exercise, e.g.: in the 1st exercise *isPrime()*. Similarly, the use of sorting and searching functions/methods is not allowed.

If during the exercise the task for pulled thesis is deemed insufficient by the committee, then the grade for the final exam is insufficient.

If you have questions, ask in an e-mail at *falucskai.janos@nye.hu*.

Theses

1. **The C programming language I.:** Data types, their declarations, conditional statements.

   General knowledge: data and information, types of entropy, infix and postfix forms of expressions.

   Artificial intelligence: state space representation of search problems, examples. Uninformed search procedures (depth, latitude, optimal).

2. **The C programming language II.:** cycle organization options, function management, parameter evaluation, scope management (static, dynamic).
   General knowledge: number systems, number representation (fixed and floating point), character, text and logical data representation.
   Artificial intelligence: The concept of heuristics, examples. The A* algorithm. The completeness of the A* algorithm. Two-player, full-information, deterministic games: the concept of strategy, minimax algorithm, alpha-beta cut.
3. **The C programming language III.:** Arrays, pointers, dynamic memory management. String management.
   Formal languages and automata: context-free grammars, CNF, CYK algorithm.
   Networks: data link protocols, layers. Local area networks. Internet basics,
   HTML.
4. **Data structures and algorithms:** Functional specification. Programming topics: search, sort, decision, selection.
   Database systems: The relational data model. Entity, attribute, relation and relationship.
   Key, foreign key, referential integrity. Constraints on database elements.
   Triggers.
   Theory of computation: Turing machines, Church thesis, halting problem, algorithmically undecidable problems. Definition of logical functions, KNF, DNF, logical networks.
   Storage and time complexity.
5. **Data structures and algorithms:** algorithms for operations on queues, stacks, incomplete matrices, linked lists, binary trees.
   Operating systems: Process management and scheduling. Memory management. File management.
   Formal languages and automata: The empty word lemma. The concept of finite automata, its types, making finite automata deterministic.
6. **Object-oriented programming:** OOP, types and their conversions, operators, statements. Methods, class creation, visibility, constructor.
   Formal languages and automata: Concepts of alphabet, word, language, grammar. Chomsky's grammatical classes and the inclusion hierarchy of language classes generated by them.
   Computer architectures: logic circuits, combinational logic networks (half and full adders, multiplexers, demultiplexers, decoders).
7. **Object-oriented programming:** Inheritance, overloading, polymorphism. Exception handling.
   Database systems: View tables in relational database managers. Indexing on tables –
   when do we use them? Database Design Theory: functional dependencies and normalization – Boyce-Codd Normal Form (BCNF). Anomalies in non-normalized database schemas. The E/K model and its translation into a relational data model.
   Programming technologies: Tracing and debugging, unit testing, logging.
   Using collections, managing relational databases in OO programming languages.
8. **Object-oriented programming:** Class and specimen initialization, constructor. Interfaces. Generic programming, classes implementing complex data structures and their most important operations.
   Systems development technology: systems development models, design, testing, UML class diagram. Version management.

Computer architectures: Fundamentals of microelectronics (semiconductors, diodes, transistors, their types and the gates they can implement). The CPU and its structure. Integrated circuits.
Types of memories, and their classification.

9. **SQL:** Data Declaration Language (DDL), the options of the CREATE TABLE and ALTER TABLE statements. Data Query Language (SELECT): sorting, filtering, grouping, multi-table queries, the difference between INNER JOIN and OUTER JOIN. Data modification (DML) sublanguage: INSERT, UPDATE, DELETE. Nested subquery options: IN, EXISTS, ALL, ANY. Linked subquery.
   Programming technologies: Design patterns in an OO programming language. MVC, like model-view-controller pattern and some other design patterns.
   Networks: Topologies and architectures. The OSI model. Physical transmission characteristics and methods, medium access methods.

Exercises

1. Write a function or method that determines whether a natural number is prime or not!
2. Write a function or method that determines whether a natural number is a perfect number or not! (the sum of its positive divisors is twice the number)
3. Write a function or method that randomly shuffles the characters in a string (random number generator can be used)!
4. Using the following approximate formula, write a function or method that returns the square root of a real number. Use the series $x_{k+1} = 1/2 * (x_k + \mathrm{a}/x_k)$, which converges to the square root when $x_1 = 1$.
5. Write a function or method that returns the cube root of a real number! Use the sequence $x_{k+1} = 1/3 * (2 * x_k + a/x_k^2)$, which converges to its cube root when $x_1 = 1$.
6. Write a function or method that calculates the nth Fibonacci number! The Fibonacci series is defined by the recursion $a_n = a_{n-2} + a_{n-1}$ (n>2), where $a_1 = a_2 = 1$.
7. Write a function or method that returns the largest integer power of a natural number that is just less than 567!
8. Write a function or method that, for a natural number, prints out how many of the digits 9 it contains (do not convert it to a string/character array)!
9. Write a function or method that determines whether the second bit from the right of a natural number in its binary representation is 1 or 0 (do not convert it to a string/character array)!
10. Write a function or method whose parameter is a natural number 1 < x < 10, and which prints the first five elements of the sequence 1,3,4,6,7,9,10,12,... that are divisible by x, i.e. the i+1th element of the sequence is 2 greater than the ith if i is odd, and 1 greater than the ith if i is even!
11. Write a function or method that returns the number of positive divisors of a natural number given as its parameter.
12. Write a function or method that removes all characters except digits from a string.
13. Write a function or method that determines whether a string is a palindrome! (Reading it from the left is the same as reading it from the right.)
14. Write a function or method that converts the first letter of each word in a string containing the letters of the English alphabet to uppercase.
15. Write a function or method that removes all occurrences of a specified character from a string.
16. Write a function or method that counts all occurrences of a given string in another string.
17. Write a function or method that prints out the letters of the English lowercase alphabet whose ASCII code is a square number!
18. Write a function or method that generates a random string of 5 characters (using lowercase English alphabet characters). Ensure that each string of 5 different letters of length is chosen with equal probability, provided that the random number generator of the chosen programming language ensures a uniform distribution!
19. Write a function or method that inserts the character "a" into a string at a randomly chosen position (a random number generator can be used)!
20. Give a function or method that, given two positive integer parameters, returns: $\binom{n}{k} = n!/(k! * (n-k)!)$. Use recursion!
21. Provide a method or function that determines whether a non-empty array (of positive integers) given as a parameter contains a number that divides all the others. (A remainder function can be used).
22. Provide a method or function that determines whether a non-empty array (consisting of positive integers) given as a parameter contains a number that occurs more times than all the others.
23. Provide a method or function that returns the index in the non-empty array (of positive integers) given as its parameter where the longest continuously increasing subsequence begins. If there are multiple such indexes, return the last one.

24. Provide a method or function that returns the smallest index in a nonempty array (consisting of positive integers) given as its parameter, at which the sum of the elements before the index exceeds the product of the first two elements of the array. If there is none, return 0.
25. Provide a method or function that, given a string/character array s, a character c, and a positive integer n as parameters, returns the nth occurrence of the character c in the string s.
26. Provide a method or function that searches for the index of the first occurrence of the integer given as a parameter in a sorted array of integers in a number of steps proportional to the logarithm of the length of the array, or if there is none, returns -1. (e.g. binary search)
27. Write a function or method that returns the second digit from the left of the product of two integer parameters! (do not use strings/character arrays in the solution)
28. Write a function or method that determines whether the 5x5 /array of characters/ received as a parameter has an element on the main diagonal that also appears in the array outside the main diagonal!
29. Write a function or method that returns the digit after the decimal point of its real-type parameter! (do not use strings/character arrays in the solution)
30. Write a function or method that returns the inverse of its positive integer parameter, e.g. invert(234) returns 432! (do not use strings/character arrays in the solution)4
31. Write a function or method that, given a 10x10 matrix as a parameter, determines whether there is an element in it that is larger in its row and smaller in its column than the other elements!
32. Write a function or method that returns how many integers there are that are greater than k, less than m, and have exactly n real divisors! (k, m, n are natural numbers, k <= m)
33. Write a function or method that returns the number of digits in the least common multiple of two positive integer parameters in binary.