Computer science

1 – 5

What is hardware?

:r1 Physical components of computer

:r2 Programs stored in computer

:r3 Unit of information

:r4 Parts of the computer which cannot be touched

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

What is software?

:r1 Physical parts of the computer

:r2 Programs

:r3 Unit of information

:r4 Part of the computer which can be touched

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

What is a positional numerical system?<br />

&nbsp;

:r1 The value of the number is given by the sum of all digits

:r2 Numerical system in which all mathematical operations can not be performed.

:r3 The value of each digit is given by its position in the symbol sequence

:r4 A method of representing numbers in which no digit is given by its location in a given sequence of digits

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

What is a non-positional numerical system?<br />

&nbsp;

:r1 The value of the number is given by the sum of all digits

:r2 Numerical system in which all mathematical operations can not be performed.

:r3 The value of each digit is given by its position in the symbol sequence

:r4 A method of representing numbers in which the value of a digit is not given by its location in a given sequence of digits

:r1 0

:r2 0

:r3 0

:r4 2 ok

--

What is logic?<br />

&nbsp;

:r1 The science of correct reasoning and art of argumentation

:r2 Prerequisite procedure to resolve the problem

:r3 Information you need to get first

:r4 The way we draw general conclusions from specific observations

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

What is induction?<br />

&nbsp;

:r1 Generalization; the procedure where we draw general conclusions from specific observations

:r2 The science of correct judgment

:r3 We create hypotheses for observed phenomena, diagnosis of "disorders"

:r4 Conclusion logically follows from the assumptions, if under no circumstances can the case arise such that the assumptions would be true and the conclusion untrue.

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

What is deduction?<br />

&nbsp;

:r1 Generalization; the procedure where we draw general conclusions from specific observations

:r2 The science of correct judgment

:r3 We create hypotheses for observed phenomena, diagnosis of "disorders"

:r4 Conclusion logically follows from the assumptions, if under no circumstances can the case arise such that the assumptions would be true and the conclusion untrue.

:r1 0

:r2 0

:r3 0

:r4 2 ok

--

What is abduction?<br />

&nbsp;

:r1 Generalization; the procedure where we draw general conclusions from specific observations

:r2 The science of correct judgment

:r3 We create hypotheses for observed phenomena, diagnosis of "disorders"

:r4 Conclusion logically follows from the assumptions, if under no circumstances can the case arise such that the assumptions would be true and the conclusion untrue.

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

What is a set?

:r1 Summary of information about the given object

:r2 Another name for the relations

:r3 Objects that can not be mathematically described

:r4 A collection of objects that are precisely defined and distinguishable and form part of the world of our ideas and thoughts;

:r1 0

:r2 0

:r3 0

:r4 2 ok

--

What is a relation?<br />

<br />

<br />

&nbsp;

:r1 Summary of information about the given object

:r2 n-ary Relations between sets an arbitrary subset of the Cartesian product of n sets

:r3 Objects that can not be mathematically described

:r4 A collection of objects that are precisely defined and distinguishable and form part of the world of our ideas and thoughts;

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

What is a potency set?<br />

<br />

<br />

&nbsp;

:r1 A set containing no element

:r2 Set of all subsets

:r3 A set of all elements

:r4 A set of all sessions

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

What is the Cartesian product X × Y?<br />

<br />

<br />

&nbsp;

:r1 A discrete product, a set comprising disordered pairs where one or both elements can be both X and Y

:r2 The set X × Y, which contains all ordered pairs, where the first entry is from X and the second entry is from Y

:r3 A set of such pairs, where the element of X is equal to the element of Y

:r4 A set of all elements of X and Y.

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

What is a targeted graph?<br />

&nbsp;

:r1 Pairs (V, E) where E is a subset of the Cartesian product V × V. E is a ordered pair of vertices

:r2 A list of vertices and edges where the edges form disordered pairs

:r3 A set where each non-empty part has the smallest element.

:r4 Draw diagram of the implementation of the algorithm

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

What is an ordered set?<br />

&nbsp;

:r1 A set where any two elements can not be compared

:r2 A set on which the relation R is defined, which is transitive, weakly antisymmetric and trichotomic.

:r3 Another name for an empty set

:r4 The set of all ordered pairs (x, y) for which x <y is true

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

What is a sharp arrangement?<br />

<br />

<br />

&nbsp;

:r1 The set is sharply arranged if all of its elements are equal.

:r2 Another name for an empty set

:r3 For every two elements of the set, $x < y$.

:r4 For every two elements of the set, $x \leqq y$.

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

What is non-sharped arrangement?

:r1 A set where any two elements can not be compared

:r2 Another name for an empty set

:r3 For every two elements of the set, $x < y$.

:r4 For every two elements of the set, $x \leqq y$.

:r1 0

:r2 0

:r3 0

:r4 2 ok

--

What is Injection (injective morphism)?<br />

&nbsp;

:r1 $f: A \to B$, the whole set B is a range of values

:r2 one to one

:r3 f: A → B and g: B → C, h: A → C, H = g∘f,

:r4 f: A → B, ∀b∈B: ∃! a ∈A: f (a) = b

:r1 0

:r2 0

:r3 0

:r4 2 ok

--

What is bijection (bijective morphism)?<br />

<br />

<br />

&nbsp;

:r1 f: A → B, the whole set B is a range of values

:r2 one to one

:r3 f: A → B and g: B → C, h: A → C, H = g∘f,

:r4 f: A → B, ∀b∈B: ∃! a ∈A: f (a) = b

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

What is a surjection (surjective morphism)?<br />

<br />

<br />

&nbsp;

:r1 f: A → B, the whole set B is a range of values

:r2 one to one

:r3 f: A → B and g: B → C, h: A → C, H = g∘f,

:r4 f: A → B, ∀b∈B: ∃! a ∈A: f (a) = b

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

What is function composition?<br />

<br />

<br />

&nbsp;

:r1 f: A → B, the whole set B is a range of values

:r2 one to one

:r3 f: A → B and g: B → C, h: A → C, H = g∘f,

:r4 f: A → B, ∀b∈B: ∃! a ∈A: f (a) = b

:r1 0

:r2 0

:r3 2 ok

:r4 0

6 – 9

Boolean algebra - choose the right statement<br />

<br />

<br />

&nbsp;

:r1 (A ⟹ B) = (NOT (A) ∨ B)

:r2 (A⟹B) = (A∨B)

:r3 (A ⟹ B) = (A ∧ NOT (B))

:r4 (A ⟹ B) = (NOT (A) ∨NOT (B))

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Boolean algebra - choose the right statement<br />

<br />

<br />

<br />

&nbsp;

:r1 (A⇔B) = (NOT (A) ∨B)

:r2 (A⇔B) = (NOT (A XOR B))

:r3 (A⇔B) = (NOT (A) ∨NOT (B))

:r4 (A⇔B) = (NOT (A) ⇔B)

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

Boolean algebra - choose the right statement<br />

<br />

<br />

&nbsp;

:r1 (A XOR B) = (NOT (A) ∨NOT (B))

:r2 (A XOR B) = (A ∧ NOT (B))

:r3 (A XOR B) = ((B∧NOT (A)) ∨ (A∧NOT (B)))

:r4 (A XOR B) = (NOT (A) ∨ (B))

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

Boolean algebra - choose the right statement<br />

<br />

<br />

<br />

&nbsp;

:r1 NOT (A ∧ B) = (NOT (A) ∨NOT (B))

:r2 NOT (A ∧ B) = (NOT (A) ∧NOT (B))

:r3 NOT (A ∧ B) = 0

:r4 NOT (A ∧ B) = 1

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

What is determinism?<br />

<br />

<br />

<br />

&nbsp;

:r1 A set of concepts, rules and procedures that help us organize numerical information

:r2 Determinism describes the behavior of the set

:r3 It describes a situation where there is only one possible future, and it is theoretically quite predictable.

:r4 Describes a situation where there are multiple possible futures, chooses among them randomly.

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

What does stochasticity mean?<br />

<br />

<br />

<br />

&nbsp;

:r1 A set of concepts, rules and procedures that help us organize numerical information

:r2 Stochasticity describes the behavior of sessions

:r3 It describes a situation where there is therefore only one possible future, and it is theoretically quite predictable.

:r4 Describes a situation where there are multiple possible futures, chooses among them randomly.

:r1 0

:r2 0

:r3 0

```
:r4 2 ok

--

What is a Modus?<br />

<br />

<br />

<br />

&nbsp;

 :r1 Average value

 :r2 The most common value

 :r3 The value of the distribution right in the middle

 :r4 The sum of the smallest and largest values divided by two

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

What is a median?<br />

<br />

<br />

<br />

<br />

&nbsp;

 :r1 Average value

 :r2 The most common value

 :r3 The value of the distribution right in the middle

 :r4 The sum of the smallest and largest values divided by two

:r1 0
```

:r2 2 ok

:r3 0

:r4 0

--

The noise channel describes<br />

<br />

<br />

<br />

&nbsp;

:r1 Transferring information using noise-free channel

:r2 Transfer information using noise

:r3 Transfer information using a noise channel

:r4 How does the channel transmit noise without transmitting a message

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

Syntax:<br />

<br />

<br />

<br />

&nbsp;

:r1 Describes the qualitative side of the information

:r2 Specifies the message priority

:r3 It is the meaning of information

:r4 It concerns the mutual arrangement of the characters as information carriers

:r1 0

:r2 0

:r3 0

:r4 2 ok

--

Semantics:<br />

<br />

<br />

&nbsp;

:r1 Describes the quantitative information page

:r2 Specifies the message priority

:r3 It is the meaning of information

:r4 It concerns the mutual arrangement of characters as information carriers

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

Pragmatic content<br />

<br />

<br />

<br />

&nbsp;

:r1 Describes the qualitative side of the information

:r2 Describes the quantitative information page

:r3 It is the meaning of information

:r4 It concerns the mutual arrangement of characters as information carriers

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Securing parity<br />

<br />

<br />

&nbsp;

:r1 It means that a new data segment is created, which determines the number of nulls and numbers in the data

:r2 This means that each segment is connected to another bit in order to complete the number of binary ones on an even (even parity) or odd (odd parity) number

:r3 The data is divided into sections of the required length (8, 16, 32 bits) and these segments are added to the bits without transfer. The resulting data segment connects to the data transmitted.

:r4 A new data segment is attached to or logically linked to a data message, enabling the person to be authenticated in relation to the data message.

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

CRC<br />

<br />

<br />

<br />

&nbsp;

:r1 It means that a new data segment is created, which determines the number of nulls and numbers in the data

:r2 This means that each segment is connected to another bit in order to complete the number of binary ones on an even (even parity) or odd (odd parity) number

:r3 The data is divided into sections of the required length (8, 16, 32 bits) and these segments are added to the bits without transfer. The resulting data segment connects to the data transmitted.

:r4 A new data segment is attached to or logically linked to a data message, enabling the person to be authenticated in relation to the data message.

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

Hamming's code<br />

<br />

<br />

<br />

&nbsp;

:r1 linear code used in telecommunications to detect up to two erroneous bits or to repair one bad bit

:r2 It means that a new data segment is created, which determines the number of nulls and numbers in the data

:r3 It is a way to express the algorithm for data transfer

:r4 The data is divided into sections of the required length (8, 16, 32 bits) and these segments are added to the bits without transfer. The resulting data segment connects to the data transmitted

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Electronic signature<br />

<br />

<br />

<br />

&nbsp;

:r1 It means that a new data segment is created, which determines the number of nulls and numbers in the data

:r2 This means that each segment is connected to another bit in order to complete the number of binary ones on an even (even parity) or odd (odd parity) number

:r3 The data is divided into sections of the required length (8, 16, 32 bits) and these segments are added to the bits without transfer. The resulting data segment connects to the data transmitted.

:r4 It is attached to, or logically associated with, a data message, allowing you to verify the identity of the signer in relation to the data message.

:r1 0

:r2 0

:r3 0

:r4 2 ok

10 – 13

complexity theory

:r1 Describes how complex it is to implement a specific algorithm

:r2 It focuses on the classification of computational problems according to their own complexity and the determination of relations between classes.

:r3 It says that if the problem is complex, it can not be solved by algorithm

:r4 It deals with description of the algorithm and its simplification

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

Analysis of algorithms

:r1 It deals with the amount of resources needed that a particular algorithm needs

:r2 He says that if the problem is complex, it can not be solved by algorithm

:r3 It deals with description of the algorithm and its simplification

:r4 It focuses on the classification of computational problems according to their own complexity and the determination of relations between classes.

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

The decision problem

:r1 It is such a problem that we have to decide at least between the three results

:r2 There is a problem in which the program always stops without any result.

:r3 the basic type of complexity problems, has only two outputs Yes / No

:r4 describes a situation where each problem can be decided maximum in linear time (n-times the input size)

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

NP-complete problems

:r1 there are such non-deterministic polynomial problems on which all other NP problems are polynomially reducible

:r2 a problem that is completelly described in Turing

:r3 are problems whose solution can be found in a deterministic Turing machine in linear time.

:r4 They can never be solved - completely unsolved problems

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Language

:r1 A set of words that can not be described by a regular expression

:r2 Structure describing grammar

:r3 A set of words that are not accepted by any automaton

:r4 It is defined by using any non-empty set V, which is an alphabet, its elements are characters or symbols

:r1 0

:r2 0

:r3 0

:r4 2 ok

--

finite state machine

:r1 It describes a very simple computer that can be in one of several states between which it switches on the symbols read from the input

:r2 It is a machine whose set of states is not final, the final machine has a final memory of at least two steps back (due to the use of the binary system)

:r3 Machine that has only two possible states, and its run is over is always the first step

:r4 A machine whose activity never ends but runs on the final set of states with a final memory whose size is equal to the number of states on the other

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Deterministic finite automaton

:r1 At each point of the transition table, it has just one target state

:r2 It is a machine whose set of states is not final, the final machine has at least two steps back (due to the use of the binary system)

:r3 There is not only one target state at each point of the table but the whole set of states

:r4 A finite slot machine, whose transition table is always a single line

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Non-deterministic finite automaton

:r1 It is a machine whose set of states is not final, the final machine has at least two steps back (due to the use of the binary system)

:r2 At each point of the transition table, it has just one target state

:r3 There is not only one target state at each point of the table but the whole set of states

:r4 Finite automaton machine, whose transition table is always a single line

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

Turing machine

:r1 A finite automaton that contains a right-handed endless tape on which it writes the states, and a left-hand endless tape from which the instruction reads

:r2 It consists of a processor unit consisting of a finite automaton, a program in the form of transition function rules, and right-edge endless tapes for writing intermediate results

:r3 It is the only type of machine (algorithm) that is able to solve NP-complete problems in linear time.

:r4 It is the only type of machine (algorithm) that can handle NP-complete problems in constant time.

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

N-tape Turing machine

:r1 It is a modification of the basic Turing machine, which allows a "choice from multiple options" ie in the transition table has n-state tables at each location

:r2 It is a modification of the basic Turing machine, reads and writes multiple tapes at once

:r3 It is a modification of the basic Turing machine that can simulate any Turing machine

:r4 Is a modification of the basic Turing machine, the machine writes the state on n-tapes and reading is always only from one

:r1 0

:r2 2 ok

:r3 0

:r4 0

--

Universal Turing machine

:r1 It is a modification of the basic Turing machine, which allows a "choice from multiple options" ie in the transition table has n-state tables at each location

:r2 It is a modification of the basic Turing machine, reads and writes multiple tapes at once

:r3 It is a modification of the basic Turing machine that can simulate any Turing machine

:r4 Is a modification of the basic Turing machine, the machine writes the state on n-tapes and reading is always only from one

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

Turing's completeness

:r1 Turing's complete machine solves NP-complete problems in constant time

:r2 Turing's complete machine can solve any problems in the final time, including a halting problem

:r3 Turing's complete are the programming languages and computers that have the same computational power as Turing's machine

:r4 Turing's complete machine solves NP-complete problems in linear time

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

computability theory<br />

&nbsp;

:r1 Basic theory of computability questions are: What is and what is not algorithmically computable (solve)? What problems do the algorithms solve?

:r2 It attempts to quantify how many algorithm steps it takes to resolve the problem.

:r3 The basic question of solving theory is: What minimal algorithm solves the problem?

:r4 Says how many resources are needed to simulate the problem using the Turing machine.

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Algorithmic solvability<br />

<br />

<br />

&nbsp;

:r1 The problem is that it is algorithmically solvable if there is an algorithm that is capable of accepting any instance of a given problem, and its calculation for any such input always ends, with output being the desired result.

:r2 We will say that it is algorithmically solvable for the given problem Yes / Ne problem if there is an algorithm capable of accepting any instance of the problem as input and its calculation for any such input always ends and the output will be the Yes answer. If the answer is not correct, then the algorithm will not end.

:r3 The problem is algorithmically solvable if we can build such an algorithm (Turing machine) that gives us the desired result in time linearly dependent on input size.

:r4 The problem is that it is algorithmically solvable if a deterministic finite automaton can be constructed to provide the right result in all circumstances irrespective of size and type of input at a time that is maximally linearly dependent on input size.

:r1 2 ok

:r2 0

:r3 0

:r4 0

--

Algorithmic decidability<br />

&nbsp;

:r1 The problem is algorithmically decidable if it can put together such an algorithm (Turing machine), which will give us the desired result in time linearly dependent on the size of the input.

:r2 The problem we say that algorithmically decidable if there is an algorithm that is able to receive as input any instance of the problem and its calculation for any such input, always results in an output of the desired result.

:r3 About the problem of Yes / No we will say that is algorithmically decidable if there is an algorithm that is able to accept as input any instance of the problem and its calculation for any such entry ever ends, the output will be required answer Yes / No.

:r4 The problem is decisive if we know the algorithm that stops and produces the result only if the answer to the question is NO. If the answer is YES, we will never know it because the algorithm will never stop.

:r1 0

:r2 0

:r3 2 ok

:r4 0

--

Partial decidability<br />

<br />

&nbsp;

:r1 The problem is partially decidable knowing the algorithm stops and gives a result only if the answer to the question Sun. If the answer is YES, we will never know it because the algorithm will never stop.

:r2 The problem we say that algorithmically partially solvable if there is an algorithm that is able to receive as input any instance of the problem and its calculation for any such input, always results in an output of the desired result.

:r3 The given Yes / No type problem is said to be partially determinable by algorithm if an algorithm is available that accepts any instance of the problem and its calculation for any such input always ends and the output will be the Yes / No response.

:r4 The problem is partially decisive if we know the algorithm that stops and generates the result only if the answer to the question is Yes. If the answer is No, so we will never know, because the algorithm never stops.

:r1 0

:r2 0

:r3 0

:r4 2 ok