1. Who is the author of the *primary* textbook for the class?
   (a) Sibelius; (b) Sebesta; (c) Sethi; (d) Stansifer; (e) Scott

2. How many students received an ‘F’ from the instructor recently for reasons of academic misconduct?
   (a) 2; (b) 20; (c) 200; (d) 2,000

3. true / false There will be a short quiz at the beginning of (almost) every class meeting.

4. In a class of 50 computer science seniors, one should expect how many to fail to put their name on the quiz card?
   (a) 0; (b) 5; (c) 10; (d) 50
1. *Abstraction* means
   (a) signification of words or forms; (b) the medium of expression;
   (c) human-oriented presentation of data; (d) free from convoluted
   interactions; (e) act of determining essential properties.

2. Arabic is to linguistics as:
   (a) expression::visualization; (b) complexity::abstraction;
   (c) Python::programming languages; (d) gender::sex.

3. Software engineers need math because:
   (a) computer operations come from mathematics;
   (b) visualization of data is mathematical; (c) calculus is essential
   to calculating; (a) software consists of abstract constructs.

4. true / false   Complexity can be harmful.

5. In the field of programming languages one studies: (a) the
    writings of Guido van Rossum; (b) expressing computation;
    (c) visualization data; (d) the LAMP stack.
Quiz: Mon, 27 Aug 2018

1. Which one of the following is *not* a computational paradigm: (a) logic programming; (b) imperative; (c) categorical; (d) functional.

2. true / false Recursion is a distinguishing characteristic of the computational paradigms.

3. true / false Objects are a distinguishing characteristic of the computational paradigms.

4. true / false Evidence of the Sapir-Whorf hypothesis is the fact that the people cling to their first programming language.

5. The sky does not have a color because: (a) it contradicts physics; (b) the ancient Greeks drank too much (purple) wine; (c) ancient people couldn’t see the sky because of thick (green) foliage; (d) one cannot paint it
1. Alfred Tarski (1902–1983) is known for: (a) theory of computation; (b) semantics; (c) finding’s Frege’s flaw; (d) theory of quantification

2. Using predicate logic, formulate the sentence: every student has some professor they admire. Use the predicates $S(x)$ for student, $P(x)$ for professor, and $A(x, y)$ for $x$ admires $y$.

3. true / false Programming on a Turing machine is very rudimentary.

4. Which one of the following does not predate FORTRAN (a) Plankalkül; (b) IBM 704; (c) APL; (d) Short code

5. Which one of the following is not a programming language: (a) Go; (b) Rust; (c) Ruby; (d) Diamond; (e) Lua
Quiz: Wed, 29 Aug 2018

1. true / false  Declarative programming emphasizes the “how” over the “why.”

2. true / false  HTML is not a general purpose programming language.

3. true / false  Russell’s paradox is a linguistic and not a mathematical paradox.

4. XXXIII divided by IV is:
   (a) V; (b) VI; (c) VII; (d) VIII; (e) IX.

5. The name of the Greek letter π rhymes with all of the following Greeks letters except for:
   (a) ν; (b) ξ; (c) φ; (d) χ; (e) ψ.

Use the back of old quiz cards!
1. true / false FORTRAN can reasonably be considered the first programming language.

2. true / false The syntax known as “Cambridge prefix” is used for all data and all code in LISP.

3. true / false ISO stands for the organization officially known as the International Standards Organization.

4. Surprisingly the programming language is used today in financial applications. It was originally designed in the 1960’s to describe hardware and relies heavily on arrays and vectors. It has a modern successor called.

5. Grace Hopper led the development of , a programming language for business applications.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th><strong>Languages</strong></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td>Ada</td>
<td></td>
<td>A.</td>
<td>IBM, J. Backus</td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td>APL</td>
<td></td>
<td>B.</td>
<td>Augusta Ada</td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>COBOL</td>
<td></td>
<td>C.</td>
<td>Dahl and Nygaard</td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td>C++</td>
<td></td>
<td>D.</td>
<td>Ralph Griswold</td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td>C</td>
<td></td>
<td>E.</td>
<td>Kenneth Iverson</td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td>FORTRAN</td>
<td></td>
<td>F.</td>
<td>Guido van Rossum</td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>Java</td>
<td></td>
<td>G.</td>
<td>John McCarthy</td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>LISP</td>
<td></td>
<td>H.</td>
<td>Kernighan and Richie</td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>Pearl</td>
<td></td>
<td>I.</td>
<td>Bjarne Stroustrup</td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td>Python</td>
<td></td>
<td>J.</td>
<td>Sun, J. Gosling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>K.</td>
<td>US DoD, G. Hopper</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L.</td>
<td>US DoD, J. Icbaih</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>M.</td>
<td>Larry Wall</td>
</tr>
</tbody>
</table>
Quiz: Wed, 7 Sept 2018

1. A *lexeme* is a
   (a) letter; (b) token; (c) word; (d) phrase

2. In formal languages, a *symbol* is
   (a) a letter used to designate something (b) hallmark or emblem
   (c) a sign to represent something such as an organization (d) one indivisible element of a notational system

3. The perspective the programming language field has on syntax can best be described as:
   (a) annoyance; (b) basic implementation; (c) construction;  
   (d) description

4. true / false Formal language theory applies to the lexical structure of programming languages, but not to the phrase structure.

5. true / false A formal language is a set of symbols from an alphabet.

6. true / false Language can be studied in three parts: pragmatics, syntax, and semiotics.
Quiz: Mon, 10 Sept 2018

What formal languages over the alphabet \{a, b, c, d\} do the following regular expression represent? Choose from the formal languages below. (You may choose a letter any number of times.)

1. \( \emptyset^* \)
2. \((a + b)^*\)
3. \((a^*)^*\)
4. \((a + \emptyset)^*\)
5. \(((a \cdot b) + (c \cdot d))\)
6. \(((a \cdot b) + (c \cdot d))^* \cdot c)\)
7. \(((a + b) + a^*) \cdot c)\)
8. \((a^* + b)^*\)
9. \(((a + b)^* + (a + c)^*)\)

A. \{\}
B. \{\epsilon\}
C. \{abcd\}
D. \{ab, cd\}
E. \{a, b, aa, ab, ba, bb, \ldots\}
F. \{\epsilon, a, b, aa, ab, ba, bb, \ldots\}
G. \{ac, bc, aac, abc, bac, bbc, \ldots\}
H. \{c, abc, cdc, abcdc, cdcdc, cdabc, ababc, \ldots\}
I. none of the above
Quiz: Wed, 12 Sept 2018

1. true / false Syntax diagrams are equivalent to context-free grammars.

2. true / false Back references can be defined in terms of the primitive regular expressions and, so, are just “macros” or “syntactic sugar.”

3. true / false Regular expressions are great because they are more expressive than other common formalisms.

4. true / false There are tools in wide-spread use to generate scanners automatically from regular expressions.

5. true / false Scanner generators and parser generators are examples of a kind of programs which enable programmers to describe what they want and not how to implement it.

6. If \(d\) is the regular expression for a digit (\([0–9]\)), then the regular expression \(d+ \ d+?\) matches what part of 321 3451324?
   (a) 3 3; (b) 3 3451324; (c) 321 3; (d) 321 3451324.
1. Show that the following grammar with non-terminals $S$, $A$, and $I$ is ambiguous:

$$
S \rightarrow A \\
A \rightarrow A \times A | I \\
I \rightarrow a | b | c
$$
1. Show that the following grammar with non-terminals $S$, $A$, and $I$ is ambiguous:

$$S \rightarrow A$$

$$A \rightarrow A \times A \mid I$$

$$I \rightarrow a \mid b \mid c$$
1. true / false  The propositional formula $A \& B \rightarrow C$ is necessarily true if $B$ is false.

2. true / false  Assertions in programs help debugging.
Quiz: Mon, 17 Sep 2018

Test on Monday, 24 Sept 2018!
Syntax and Semantics

1. true / false The Cherokee script is used in writing FORTRAN programs.
2. true / false Assertions in programs help debugging.
3. true / false A formula of first-order logic can be used to characterize a set of computer states.
4. true / false “Sue me if my postcondition is false, sue you if my precondition is false.”
5. Complete the last two columns of the following truth table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A&amp;B</th>
<th>A&amp;B ⇒ C</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
Quiz: Wed, 19 Sep 2018

Test on Monday, 24 Sept 2018!
Syntax and Semantics

Fill in the box with the phrase that best describes the approach of each of the following types of semantics:

1. denotational
2. operational
3. natural
4. structural
5. axiomatic

is defined in terms of . . .

A rules for evaluation
B Post systems
C rules relating states
D attribute grammars
E mathematical objects
F “small-step” transitions
G an abstract machine
H a 1985 song by Paul Hardcastle “19”
1. true / false  An inference rule is a method of asserting the truth of one assertion on the basis of the form of other assertions.

2. true / false  Partial correctness means the program satisfies some of the postconditions.
Using the axiom of assignment and the rule of composition (but not the rule of consequence), compute the (weakest) precondition for the following program fragments with respect to the given postcondition.

1. \( a := b + 3 \{ a = 7 \} \)
2. \( a := a + 1 \{ a = a + b \} \)
3. \( a := a + 1 \{ b = 7 \} \)
4. \( a := a + 1; b := b \times c \{ b = c^a \} \)
Quiz: Fri, 28 Sep 2018

1. true / false Names allows programmers to refer to high-level abstractions.

2. true / false In Java every identifier must be declared before it is used.

3. true / false The variable Kotlin does not have to declared before it is used in Fortran.

4. Who was the first Turing Award recipient? (a) Alan Mathison Turing; (b) Donald Knuth; (c) Alan Jay Perlis; (d) Alan Curtis Kay.

5. What does abstraction in the context of statements lead to? (a) a head ache; (b) subprocedures; (c) functions; (d) data abstraction; (e) garbage collection.

6. What does abstraction in the context of expressions lead to? (a) another head ache; (b) subprocedures; (c) functions; (d) data abstraction; (e) garbage collection.
Quiz: Mon, 1 Oct 2018

1. true / false  
   Names refer only to locations in programs.

2. true / false  
   Late binding in generally more flexible.

3. true / false  
   An environment is a kind of a function.

4. true / false  
   In some languages, for instance Pascal, constants are required to have a value than can be determined a compile time.

5. What is the proof rule in Hoare logic for the while statement in the simple while language?
1. true / false   Extent is measured in inches and scope in time.

2. Which one of the following is not a principle storage management mechanism:
   (a) static; (b) stack; (c) instance; (d) heap

3. true / false   Elaboration time is subsumed (it occurs during) run time.

4. true / false   Stack-allocated objects require complex storage management.

5. Which one of the following programming languages does not have explicit deallocation:
   (a) C; (b) C++; (c) Java; (d) Rust
1. true / false  Pointers limit concurrency.
2. true / false  Pointers are easy to reason about.
3. true / false  Recursion is easy to reason about.
4. Which of the following is *not* a kind of assignment. (a) “let”; (b) storage; (c) pointer; (d) “with”
5. When is the Go project due?
1. true / false Conservative garbage collection may create memory leaks.

2. true / false Dynamically allocated objects tend to live a short amount.

3. true / false The primary problem with reference counting is unpredictability.

4. Which of the following is not a part of generational garbage collection. (a) eden; (b) primogeniture; (c) tenured;
1. true / false  In some languages, for instance Modula-3 and C#, *every* use of an enumeration constant must be prefixed with a type name.

2. true / false  Sophisticated alias analysis algorithms have allowed C compilers to rival Fortran compilers in the speed of generate code.

3. true / false  In the buddy system the storage management algorithm maintains separate free lists for block of different sizes.

4. true / false  Internal fragmentation is waste within a block, and external fragmentation is non-contiguous free blocks.

5. true / false  In some languages, for instance Pascal, constants are required to have a value than can be determined at compile time.

6. true / false  In C# compile-time and elaboration time constants are distinguished by using the `const` and `readonly` keywords, respectively. Java’s keyword `final` is like C# keyword `const`.

7. true / false  The variable Kotlin does not have to declared before it is used in Fortran.
Quiz: Fri, 12 Oct 2018

1. mark-sweep
2. stop and copy
3. generational
4. conservative
5. reference counting

A. Eden, tenured, . . .
B. incremental bookkeeping
C. polymorphism
D. if it looks like a pointer, . . .
E. the Korean reality girl group TV show, *Sixteen*
F. theoretical framework
G. follow everything
H. keep half in reserve
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true / false</td>
<td>Detecting errors in a program by the compiler is valuable.</td>
</tr>
<tr>
<td>2</td>
<td>true / false</td>
<td>It is possible for a compiler to reject all programs that go into an infinite loop on some input data.</td>
</tr>
<tr>
<td>3</td>
<td>true / false</td>
<td>A type insecurity arises when the data is misinterpreted.</td>
</tr>
<tr>
<td>4</td>
<td>true / false</td>
<td>Tractable means practical.</td>
</tr>
<tr>
<td>5</td>
<td>true / false</td>
<td>Early error detection reduces programming flexibility.</td>
</tr>
<tr>
<td>6</td>
<td>true / false</td>
<td>A program that cannot be statically type checked has a type insecurity.</td>
</tr>
</tbody>
</table>
How does union in the C programming language cause a type insecurity?
Quiz: Wed, 11 Oct 2017

1. What is the difference between a *statement* and an *expression*?
2. How do you parse \( a+b*c**d**e/f \) in Fortran? (Put in the missing parentheses.)
3. What error do you get in Ada with the previous expression?
4. true / false Referential transparency is a property of *language*.
5. true / false Any binary file can be interpreted as a US-ASCII text file.

Questions to think about:

1. What does it mean for a language to be expression-oriented?
2. What distinguishes *operators* from other sorts of functions?
3. Define *orthogonality* in the context of programming language design.
1. true / false An associative array is dynamically-accessed, homogeneous, composite data structure.
2. true / false A language with associative arrays can be strongly types.
3. true / false Defining the right type system is an important research area in programming language design.
4. true / false Java and C++ have evolved toward the goal of type completeness.

Questions to think about:
1. What is the difference between discrete and scalar types?
2. What are aggregates?
3. Can a language be strongly typed and dynamically typed?
1. true / false If the program misinterprets the bits, it is called a *insecurity*.

2. true / false Universal reference types (or untyped points) as in PL/I cannot be (statically) type checked.

3. true / false Variant records cannot be (statically) type checked.

4. true / false A change of type that does not alter the underlying bits is called a *nonconverting type cast* or *type pun*.

5. true / false The typing rule in Java for arrays is the same as that for ArrayList.

6. true / false If a program does not (statically) type check, then the program misinterprets the bits.

Questions to think about:

1. When are two types the same?
1. true / false In our context, polymorphism means “many values.”
2. true / false Ada uses name equivalence.
3. true / false C uses name equivalence.
4. true / false Modula-3 uses name equivalence.
5. true / false Java uses name equivalence.
6. true / false A characteristic of universal polymorphism is a finite number of possibilities.
7. true / false Implicit coercion is an example of universal polymorphism.
8. true / false “Branding” allows the programmer to make structural equivalence when the language supports name equivalence.
Quiz: Fri, 20 Oct 2017

1. true / false The record \{a:int\} is a subtype of \{a:int,b:char\}.

2. true / false Ada has inheritance as in OO languages using tagged record types.

3. true / false Implicit coercion is an example of universal polymorphism.

4. true / false Nominal subtyping means subtyping based on the programmer’s declaration of the type.

Questions to think about:

1. What is contravariance?

2. What is bounded quantification?

3. Suppose you work for IBM Rational and you are asked to implement a new feature in the IDE for Ada: a check for buffer overruns. What do you say?
Quiz: Wed, 24 Oct 2018

1. true / false The record `{a:int}` is a subtype of `{a:int,b:char}`.

2. true / false The array type operator in Java is covariant.

3. true / false The array type operator in C# is covariant.

4. true / false The `List` type operator in Java is covariant.

5. true / false The `List` type operator in Java is contravariant.

6. true / false Arrow (function) types are, by their nature, covariant in the domain and contravariant in the range.

7. true / false The typing rule for arrays in Java is *not* type safe.

8. Bounded quantification polymorphism is a combination of ______ and ______.
Quiz: Mon, 5 Nov 2018

1. true / false  Scheme is LISP cleaned up.
2. true / false  Infinite data structures are possible in eager languages.
3. true / false  Referential transparency enables equational reasoning.
4. true / false  Imperative programming is characterized by assignment, conditionals, and loops (gotos).
5. true / false  Functional programming is characterized by Cambridge prefix notation.
6. true / false  Functional programming is slow because it is usually interpreted.
7. true / false  Cambridge prefix notation cannot be used to expression assignment or loops.
8. true / false  Learn You a Haskell for Great Good.
Quiz: Wed, 7 Nov 2018

1. true / false   Your name was on the quiz card last time.
2. true / false   Strings in Haskell are arrays of characters, like in Ada.
3. true / false   Haskell programs are all written interactively.
4. true / false   Haskell is an interpreted language.
5. true / false   Haskell is named after logician Haskell Curry.

6. Arrays are to Fortran as [ ] are to Haskell.

7. What is the type of the following Haskell expression:
   (’A’, (), (’B’, ’C’), 5)

8. What is the type of the following Haskell expression:
   [ (’A’, 5), (’B’, 6) ]
Evaluate the following Haskell expressions:

1. 1:2:[]
2. [1,2+3,4]
3. 'a':['z']
4. head [1,2,3]
5. tail [1,2,3]
6. init "99.44%"
7. last [1,2,3]
8. [2,4,6]++[1,3,5]
9. []:[]
10. (!!) "Der Schuhu und die fliegende Prinzessin" 4
Quiz: Wed, 14 Nov 2018

What is type of the following Haskell expressions (for simplicity assume that all numbers have type \texttt{Int} or that $(+): : \texttt{Int} \rightarrow \texttt{Int} \rightarrow \texttt{Int}$.)

1. \( \backslash x \rightarrow x + 1 \)
2. \((+2)\)
3. \((+2)\ 5\)
4. \((2*)\)
5. \((++"z")\)
6. \("a"++).(++"z")\)
7. \texttt{let x=3 in \ \backslash y \rightarrow x \ end}\)
8. \texttt{let x=(+2) in x.x end}\)

Presentation of Awards
Quiz: Fri, 16 Nov 2018

Which of the following are patterns:

1. yes / no  x:xs
2. yes / no  [sobremesa,waldeinsamkeit,toska]
3. yes / no  x:x:xs
4. yes / no  3:xs
5. yes / no  (3:xs,’a’,_)
6. yes / no  _:xs
7. yes / no  head xs
8. yes / no  2
9. yes / no  2+3
10. yes / no  xs++ys
Quiz: Mon, 19 Nov 2018

1. What is the best type of the Haskell function \texttt{uncurry}:

\texttt{uncurry \ f \ x \ y = \ f \ (x, y)}

A. \((a \to a \to c) \to ((a,a) \to a)\)
B. \(((a, b) \to c) \to a \to b \to c\)
C. \((a \to b) \to (a,b) \to b\)
D. \((a \to b \to c) \to ((a,b) \to c)\)
E. \((a \to b \to c) \to (a,b) \to c\)

2. What is the best type of the Haskell function \texttt{map}:

\texttt{map \ f \ [] = []}
\texttt{map \ f \ (x:xs) = (f \ x) : (\texttt{map} \ f \ xs)}

A. \((a \to a) \to [a] \to [a]\)
B. \(b \to a \to [a] \to [b]\)
C. \((a \to b) \to [a] \to [b]\)
D. \(a \to b \to [a] \to [b]\)
E. \((b \to a) \to [a] \to [b]\)
3. What is the best type of the Haskell function `fold`:

\[
\text{fold } f \text{ } v \text{ } [] = v \\
\text{fold } f \text{ } v \text{ } (x:xs) = f \text{ } x \text{ } (\text{fold } f \text{ } v \text{ } xs)
\]

A. \((a \rightarrow a \rightarrow a) \rightarrow [a] \rightarrow [a]\)  
B. \((b \rightarrow a \rightarrow b) \rightarrow a \rightarrow [a] \rightarrow b\)  
C. \((b \rightarrow a \rightarrow b) \rightarrow a \rightarrow [b] \rightarrow a\)  
D. \((a \rightarrow b \rightarrow b) \rightarrow b \rightarrow [a] \rightarrow b\)  
E. \((b \rightarrow a \rightarrow b) \rightarrow b \rightarrow [a] \rightarrow b\)
Quiz: Wed, 28 Nov 2018

1. Declarative programming emphasizes:
   A. “why not” over “why”
   B. “what” over “how”
   C. “what” over “where”
   D. “how” over “what”

2. The quintessential programming language in the logic programming paradigm is:
   (A) APL; (B) Haskell; (C) Gödel; (D) PROLOG; (E) SETL.

3. We will study in this class the algorithms for:
   A. reassertion and invocation
   B. resumption and olfaction
   C. resolution and unification
   D. recession and infixation

4. true / false Logic investigates and classifies the structure of statements and arguments.

5. true / false A literal can be both a fact and a query in Prolog.

6. true / false There are no implementations of Prolog.

7. true / false Conjunction is denoted by a comma in Prolog.

8. true / false Negation is denoted by a ! in Prolog.
1. The instructor believes that being a Prolog programmer is not important. Why?

2. The instructor (and the instructor’s notes) use non-standard syntax. Why?
Quiz: Fri, 30 Nov 2018

1. true / false  A Prolog program consists of facts and queries.
2. true / false  The *closed-world assumption* states that lack of knowledge does not imply falsity.
3. true / false  Data structures in Prolog are constructed with what are called *functors* in Prolog.
4. true / false  Any computable function can be computed in Prolog.
5. true / false  Iterative-like computations can be implemented by means of recursive predicates.
6. true / false  There is no way to express “A or B is true” in the logic of Prolog.
7. The generic declarative sentence (S) is divided into a noun phrase (NP) and verb phrase (VP). The *predicate* is the [ ].
   You choices are: (S), (NP), or (VP).
1. true / false  A different query gives rise to a different Prolog search space.

2. true / false  Prolog searches the rules in order, because that is more efficient.

3. true / false  Prolog uses a depth-first search (DFS) when searching for a solution.

4. true / false  The order of the rules in a Prolog program is a factor in determining the number of solutions in the search space.

5. true / false  A Prolog search space may have an infinite number of solutions.

6. true / false  Prolog first creates the search space and then searches it.

7. true / false  A breadth-first search (BFS) of a tree requires more space than a depth-first search (DFS).
Quiz: Wed, 5 Dec 2018

1. true / false  Prolog is Turing complete.
2. true / false  Prolog uses arrays for complex data structures.
3. true / false  A Prolog query may have an infinite number of solutions, yet Prolog may find none of them.
4. Convert \([A, B, x, D]\) to an object in Prolog without using the special bracket syntax and without infix operators.
Quiz: Wed, 7 Dec 2018

1. true / false  Today is the last lecture.
2. true / false  The final exam is Thursday, 13 Dec 2018 at 8am-10am in this room.
3. true / false  Prolog is a satisfactory theorem prover.
4. true / false  Functors have to be declared before they are used in Prolog.
5. true / false  A Prolog implementation requires garbage collection.
6. true / false  Mutable collections, like arrays are important in Prolog programming.