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 45 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) 25  
5. What is the plural of the word octopus?  
   (a) octpuses; (b) octopodes; (c) octopi; (d) octpus  

Know the meaning of the words analogy, pedantic, paradigm, linguistics, i.a.
Quiz 2: Fri, 23 August 2019

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; (d) software consists of abstract constructs.

4. Sapir-Worh hypthosis states:
   (a) learning new programming languages is hard; (b) the medium constrains thought; (c) “don’t repeat yourself” (DRY); (d) all computational models are all the same.

5. In the field of programming languages one studies:
   (a) the writings of Guido van Rossum; (b) expressing computation; (c) visualizing data; (d) the LAMP stack.
1. true / false  Frege contributed to the mathematical foundations of the theory of quantification.

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

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

4. Which one of these notational systems do not violate the “arrow of time”? (a) Thai language script; (b) Peano’s notation; (c) Frege’s notation; (d) Incan quipu

5. Alfred Tarski (1902–1983) is known for: (a) theory of computation; (b) semantics; (c) finding Frege’s flaw; (d) theory of quantification
Quiz 4: Wed, 28 August 2019

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

2. true / false Translation to native code can be done by an interactive system.

3. true / false Translation to native code can be done after execution begins.

4. true / false Java cannot be translated once and then executed over and over.

5. true / false An interactive language system is always an interpreter.

6. true / false Translation to byte-code for an abstract machine is becoming more popular.

7. true / false FORTRAN can reasonably be considered the first programming language.

8. true / false The instructions of a high-level language are executed directly by the hardware.
1. Ada
2. APL
3. COBOL
4. C
5. FORTRAN
6. Java
7. Simula
8. SNOBOL

A. IBM, J. Backus
B. Augusta Ada Byron
C. Dahl and Nygaard
D. Ralph Griswold
E. Kenneth Iverson
F. Guido van Rossum
G. John McCarthy
H. Kernighan and Richie
I. Sun, J. Gosling
J. US DoD, G. Hopper
K. US DoD, J. Icbaih
L. Larry Wall
Quiz 6: Fri, 6 Sept 2019

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 7: Mon, 9 Sept 2019

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

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

3. 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.

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

5. true / false Only finite formal languages can be described by BNF.

6. true / false A BNF definition defines a formal language by providing the means to construct all and only those strings in it.
Quiz 9: Friday, 13 Sep 2019

Note: Fall Career Expo September 26 & 27 in the Clemente Center.
Note: Registration survey: https://forms.gle/d4Xnhx8nhzxpmmmdN9.

1. When is the Go project due?
2. true / false Triskaidekaphobia?
3. What does it mean for a grammar to be ambiguous?
4. Show that the following grammar with non-terminals $S$, $A$, and $I$ is ambiguous:

\[
\begin{align*}
S & ::= A \\
A & ::= A \times A | I \\
I & ::= a | b | c
\end{align*}
\]
Test on Friday, 27 Sept 2019!
Syntax and Semantics

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. denotational</td>
<td>is defined in terms of . . .</td>
<td>A</td>
</tr>
<tr>
<td>2. operational</td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>3. natural</td>
<td></td>
<td>C</td>
</tr>
<tr>
<td>4. structural</td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>5. axiomatic</td>
<td></td>
<td>E</td>
</tr>
</tbody>
</table>

- 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 quinceañera
Quiz 11: Wed, 18 Sep 2019

Assuming $x$, $y$, and $z$ are well-formed names/variables, identify which of the things below are well-formed Hoare triples, valid Hoare triples, or neither.

1. not / wff HT / valid
   \[\{ x > y \} \ x := 7 \ \{ x > y \}\]

2. not / wff HT / valid
   \[\{ 7 = 7 \} \ x := 7 \ \{ 7 = x \}\]

3. not / wff HT / valid
   \[\{ 7 = 7 \} \ x = 7 \ \{ x := 7 \}\]

4. not / wff HT / valid
   \[\{ 7 \} \ x := 7 \ \{ x = 7 \}\]

5. not / wff HT / valid
   \[\{ z = 7 \} \ x := 7 \ \{ z = 7 \}\]

6. not / wff HT / valid
   \[\{ 7 = 7 \} \ x := 7 \ \{ x := 7 \}\]

7. not / wff HT / valid
   \[\{ 7 + 1 \} \ x := 7 \ \{ x + 1 \}\]

8. not / wff HT / valid
   \[\{ \top \} \ x := 7, 8 \ \{ x = 8 \}\]

9. not / wff HT / valid
   \[\{ 7 = 7 + 1 \} \ x := 7 \ \{ x = x + 1 \}\]

10. not / wff HT / valid
    \[\{ 8 = 8 + 1 \} \ x := x + 1 \ \{ x = 8 \}\]

11. not / wff HT / valid
    \[\{ x + 1 = 3 \} \ y := 3 ; \ x := x + 1 \ \{ x = y \}\]

12. not / wff HT / valid
    \[\{ \exists x \ x > 0 \} \ y := 3 \ \{ x > 0 \}\]
1. true / false  Late binding is generally more flexible.
2. true / false  In most programming languages every identifier has an l-value.
3. true / false  In some programming languages a function may return an l-value.
4. true / false  An identifier is a variable.
5. true / false  An environment is a kind of a function.
Why Rust?
Quiz 14: Wed, 25 Sep 2019

1. true / false  Extent is measured in inches and scope in time.

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

3. Which of the following is not a kind of assignment: (a) let; (b) storage; (c) pointer; (d) with.

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

5. Which one of the following programming languages does not have deallocation of memory by the programmer:
   (a) C; (b) C++; (c) Java; (d) Pascal.
Quiz 15: Wed, 2 Oct 2019

1. What is an *expression*?
2. Define *referentially transparent*.
3. What is the significance of *referentially transparency*?
4. What is “short-circuit” evaluation?
1. A significant challenge in programming language design is:
   (a) avoiding over specification; (b) base 16 representation of
   floating-point numbers; (c) computing the correct answers; (d)
   deleting garbage.

2. A list comprehension is a programming language
(a) statement (b) expression (c) construct for understanding (d)
declaration

3. true / false In Dijkstra’s guarded command language the if fi
construction can abort the program.

4. true / false The syntax of switch statement, as in the C
programming language, has been a source of many errors.

5. true / false Where applicable, the switch statement can result
in more efficient execution than the use of if
statements.
1. true / false  Detecting errors in a program by the compiler is valuable.

2. true / false  It is possible for a compiler to reject all programs that go into an infinite loop on some input data.

3. true / false  A type insecurity arises when the data is misinterpreted.

4. true / false  The word “tractable” means practical.

5. true / false  The early detection of errors reduces programming flexibility.

6. true / false  A program that cannot be statically type–checked has a type insecurity.
Quiz 18: Wed, 9 Oct 2019

1. true / false  A pointer to anything makes static type-checking impossible.

2. true / false  An Ada subtype gives the programmer a compile-time guarantee about the behavior of the program.

3. true / false  It is possible to statically type-check heterogeneous composite data types with dynamic access.

4. What comes to mind when thinking about the implementation of variant records: (a) Ada; (b) sum; (c) average; (d) max.

5. What kind of language is Rust? (a) algebraic; (b) systems; (c) scripting; (d) functional.
1. true / false Ada uses name equivalence.
2. true / false Modula-3 uses name equivalence.
3. true / false Java uses name equivalence.
4. true / false Rust was developed by people at Google.
5. true / false One of two major design goals of Rust was to avoid the security exploits caused by the design of C and C++.
6. true / false Rust permits code that is not type safe.
1. C checks types at _______ and type safe.

2. Python checks types at _______ and type safe.

3. Rust checks types at _______ and type safe.

4. true / false If you have experience with C, C++, Java, or JavaScript, then the syntax of Rust should be familiar to you.
1. true / false A Rust enumerated type allows each variant to carry a distinct set of data values along with it.

2. true / false A Rust program will not crash because of dereferencing a null pointer.

3. true / false A Rust program will never use a heap-allocated value after it has been freed.

4. true / false Arrays are naturally contravariant.

5. true / false Function types are covariant in the domain types.
1. true / false A subtype of a function type must be a function type.
2. true / false The List type operator in Java is covariant.
3. true / false The List type operator in Java is contravariant.
4. true / false Rust’s borrow checking is carried out at compile time.
5. true / false Rust’s borrow checking is inevitably a conservative approximation.
6. Bounded quantification polymorphism is a combination of \[\_\] and \[\_\].
What is the output of the following program Algol-like program assuming dynamic scoping and assuming static scoping?

declare
  N: Integer;
procedure First is
begin
  N := 1;
end First;
procedure Second is
  N: Integer;
begin
  First;  -- call first
end Second;
begin
  N := 2;
  Second;  -- call second
  Integer_Text_IO.Put (Item=>N);
end;
1. true / false   Passing a parameter by reference generally takes less execution time than passing a parameter by value.

2. true / false   Copy in/copy out gives the same result as passing a parameter by reference.

3. true / false   One important aspect of object-oriented programming is subtype polymorphism.

4. Parnas’ two maxims of modularity are: ___________________________ and ___________________________.

1. true / false  A transparent abstract data type is one in which the representation of the type is accessible to the client.

2. true / false  A package can be stateful in Ada.

3. true / false  An abstract data type is data structure with a complete set of relevant operations.

4. true / false  One strategy to maximize cohesion in a module is to make it an abstract data type.

5. true / false  A class is a runtime mechanism and a module is a compile-time mechanism.

6. true / false  Pointers are a useful trick to isolate the implementation from the representation of a data structure.
1. What is difference between overloading and overriding?
2. What is the purpose of exception handling?
Quiz

- Fri, 8 Nov 2019. No class (SER 2019).
1. true / false  Haskell is a lazy language.
2. true / false  Lisp, Scheme, Racket, and ML are lazy languages.
3. true / false  Infinite data structures are possible in eager languages.
4. true / false  Referential transparency enables equational reasoning.
5. true / false  Imperative programming is characterized by assignment, conditionals, and loops (gotos).
6. true / false  Functional programming is characterized by Cambridge prefix notation.
7. true / false  A canonical value is one which cannot be rewritten.
8. true / false  The Glasgow Haskell Compiler has an interactive interface.
On canvas.
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
Assuming
\[ xs = [1, 2, 3, 4, 5] \]

1. What is the value of the Haskell expression
\[ xs ++ xs \]

2. What is the value of the Haskell expression
\[ (\lambda y \rightarrow xs ++ [y]) 9 \]

3. Find three things wrong with the following function definition.
\[ f (x, y) = x+y+1 \]
\[ f (_:2:y:rest) = _+2+y+1 \]
\[ f [x, y, y, z] = x+y+z+1 \]
1. What is the best type of the Haskell function `map`:

   ```haskell
   map f [] = []
   map f (x:xs) = (f x) : (map f xs)
   ```

   A. `(a -> a) -> [a] -> [a]`
   B. `b -> a -> [a] -> [b]`
   C. `(a -> b) -> [a] -> [b]`
   D. `a -> b -> [a] -> [b]`
   E. `(b -> a) -> [a] -> [b]`
   F. `[a] -> [b]`

2. What is the best type of the Haskell function `uncurry`:

   ```haskell
   uncurry f (x,y) = f x y
   ```

   A. `a -> b -> c`
   B. `(a -> b) -> c`
   C. `(a -> a) -> ((a,a) -> a)`
   D. `(a -> b) -> (a,b) -> b`
   E. `(a -> b -> c) -> (a,b) -> c`
   F. `((a -> b -> c) -> (a,b)) -> c`
Consider the following Haskell datatype definition:

```
data T a = A | B Integer | C a Integer
```

1. true / false   B is a (value) constructor.
2. true / false   T is a (value) constructor.
3. true / false   B 2 is a value of type T Integer.
4. true / false   B 2 ’a’ is a value of type T Char.
5. true / false   C 2 is a value of type Integer -> T Integer.
6. true / false   C ’a’ 2 is a value of type T Char.
7. true / false   C 2 2 is a value of type T Char.
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 A literal can be both a fact and a query in Prolog.

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

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