CSE 320 Final Exam Study Guide

(Chapter 1 – 6 will not be on test)

Chapter 1 - 4	
Is pattern matching based on name or position?	Pattern matching is based on position. The match is based on shape/type of data.
What is an "open term" in the lambda calculus?	Open term in lambda calculus is a term which is not defined in a lambda abstraction Λx.x (lambda abstraction) Λx.xy y is an open term because it's not defined. X is defined.
What are the "semantics" of a programming language?	Semantics of a programming language is the meaning in the context of the language.
What is "de-sugaring"?	De-sugaring is the process of simplifying syntactic sugar. For loops are syntactic sugar of a certain while loop. Benefits: makes writing compilers easier.
Why doesn't the omega combinator terminate? **((Λ f . (f f)) (Λ f . (f f)))**	((Λf.(f f)) (Λ f . (f f)) omega combinator \f = f f in cpp f = f(f) Let x = (\f = f f) it applies itself (almost) recursively without stopping (x x)
Optional questions: **1. What, in one sentence each, do lexer and parser do?** **2. What is an abstract syntax tree?**	1. Lexer creates tokens. A parser creates an abstract syntax tee from the tokens. 2. An abstract syntax tree is a tree data structure of the syntax.
Chapter 5 - 6	
What are the three parts of a function definition?	Name, parameters (arguments), and body.
What do we call replacing a name in an expression with another expression?	Substitution
What does an environment do?	An environment maps identifiers to values
What is static and dynamic scope?	Differences - Dynamic variables are bound during execution code. Static scope: (compile time) variables are only accessible in the scope where they were defined.
What is deep and shallow binding?	Deep binding: traverse stack if an identifier's value exists. Program stack tracks values of identifiers. Shallow binding: interpreter manually tracks values of identifiers. Deep binding and shallow binding are ways to implement dynamic scoping. (see https://www.cs.bgu.ac.il/~comp131/wiki.files/ps6.pdf)
Optional Questions: **1. What does REPL stand for?** **2. What are syntax and semantics?**	1. Read-Evaluate-Print-Loop. 2. Syntax are the rules and text for how a program is written. Semantics are the meaning of your code.

Chapter 7 (Quiz #3)	
What are closures? How are they implemented?	Closures are functions bundled with an environment. Closures are implemented with pointers to function code and the environment.
Do functions need a name? Can functions work without names?	Functions do not need a name, and yes, they can work without them (ex. lambda).
How do functions and closures relate? Are functions closures? Are closures functions?	Closures are functions with an environment. All closures are functions but not all functions are closures.
What is the "top-level" of a program?	The top level of a program refers to parts of programs that are not in the main function. Functions are not "closed over" by any identifiers - infinite scope. They are not enclose and have special rules, top level of program, global.
What is capture-free substitution? Why is it necessary?	Capture free substitution consistently renames all bound identifiers to previously unused names. User to prevent unexpected behaviors. Makes sure that values are not changed.
What is let? How is it defined?	Let is a local naming mechanism defined by de-sugaring. Let defines sugar for particular lambdas. We can assume something is defined if it's in 'let'.
Extra credit question: **1. What does it mean for a variable to be bound?** **2. What does it mean for a variable to be free?** **3. Can a variable be both "bound" and "free" in the same program?**	1. Bound – defined in the environment. 2. Free – not defined in the environment. 3. Yes. A variable can be bound and free, depends on scope.
Chapter 8 (Quiz #4)	
What is a box? What is it used for?	A box is a variable with state. A box is a field which wraps a value in a mutable container. [see photo of 5 in box being pointed to]. A structure which wraps a value and adds state.
Why does adding sequencing require the interpreter to take and return the environment?	To track the environment and pass it around. Because if we do not sent/return the environment we would not get the desired output.
What is the "store," and why is it needed? How does adding the store change the environment?	A store is a data structure which maps locations to values. Needed to keep track of states (tracks changes). A store is a data structure which keeps track of dynamic mutable values. Environment maps identifiers to locations. Store maps locations to values. State is persistent, everything that's been defined. Environment tells you what's accessible from a certain location of a program. Without store, environment maps identifiers to values.
How does adding state change the semantics of existing operations, like addition and multiplication?	Adding state, changes the order in which things are evaluated matters. When you add state to a program then you have to care about order of operations. More features you add to a language, the less assumptions you can make. Ex. In Haskell, if a function doesn't say it's using input/output, then it's not doing it. Adding state makes the way (order) things are processed matter. Ex. Haskell does not have state

What is the difference between identifiers and variables?	Identifies cannot change in scope. Names map to values. Variables are able to change in scope. (identifiers with state) values can change. Haskell identifiers can be given state -> variables
What are "call by value" and "call by reference?" What is the difference between them?	Call by value: copy the value (a copy is passed.) Call by reference: pass address (reference to address of value is passed). Changes done with call by reference can affect the original value.
How does adding the store change the environment?	The environment maps names to values. With the store, it maps names to locations which map to values.
Chapter 9 (Quiz #5)	
What is a recursive data structure?	A recursive data structure is a data structure which references itself (ex. Traversing a tree is done recursively).
What is a cyclic data structure?	A cyclic data structure is cyclic and never terminates. A cyclic data structure is a data structure which always returns to its beginning (ex. A graph is cyclic).
What does it mean for a computation to *diverge*?	A function gets a divergent type, when the function never returns. Never returns/never finishes. A computation that diverges never terminates (ex. Infinite loop).
Why does creating a cyclic datum require a box?	It requires a box because it needs a place in memory where it can reference itself. Something can't reference itself if it doesn't have a location. Boxes have a location and set the value inside the box to be that location. Can't do it without a box.
Why can't you write recursive functions without boxing?	Needs a name and has to exist (same as previous). Because binding does not make them (functions) cyclic.
Optional Questions: 1. will-stop? Is a function that attempts to solve something known as the halting problem. Why doesn't will-stop? work? 2. What does the Y (the y-combinator) do? data List a = Node a | Cons a (List a)	1. will-stop? Doesn't/cannot know whether a function will continue forever or terminate. 2. The y-combinator is a recursive function (aka magic)
Chapter 10 (Quiz #6)	
What is the relationship between functions and objects?	Objects have names, fields, and methods. An object is a value that contains other values (fields) and methods (functions)
Why objects require mutation and recursion to implement?	Because they need them for self (or this)
How objects require mutation and recursion, to enable access to self.	Together they enable objects to reference themselves. Mutation and recursion facilitate the self/this reference to the object to which those parts of the objects…
What are the design axes for names of objects?	One dimension is whether the name is provided statically or computed, and the other is whether the set of names is fixed or variable:

	Name is Static	Name is Computed
Fixed Set of Members	As in base Java.	As in Java with reflection to compute the name.
Variable Set of Members	Difficult to envision (what use would it be?).	Most scripting languages.

How languages can choose to treat names of objects.	
Quiz #7	
What is the difference between classes and prototypes?	Prototypes – same original object, children maintain pointer to parent. Each object references the object its based off. Classes define a description of an object. Which the language builds. Prototypes already have an original object where all objects are built off that. Classes are definitions which define a new object. Prototypes are objects which are based on the original object.
Why is multiple inheritance considered a bad idea?	Because of how objects would be structured (ex. Tree) looking them up would be difficult. Multiple inheritance is considered a bad idea because accessing a method which has multiple definitions can be difficult. Would need an algorithm which will defined which method to access.
What is the difference between *replacing* and *refining* when considering inheritance?	When we invoke a method, if we access it from the bottom up, we replace the parent. If we access it from the top down, we refine the parent.
What are mixins? How do they differ from inheritance?	Mixins separate extensions from base class. Mixins provide the benefits of multiple inheritance in a single inheritance language.
How do mixins differ from traits?	Traits are generalizations of mixins which extend a set of mixins. Traits are generalizations of mixins which extend a subset of mixins.
Chapter 11 (Quiz #8)	
What does it mean for memory management to be complete and sound?	When memory management is complete and sound, memory is not freed too soon (soundness) or too late (completeness).
What is fragmentation?	Fragmentation occurs when there are small pieces of memory throughout memory/ allocated memory. Holes left in storage due to freeing anything but the most recently allocated value.
What is a free-list?	A free list is a linked list of available memory.
What does it mean to trade space for time?	We maintain free lists of memory which are the same size. They are indexed and easily accessible because of this. We take up more space in exchange for accessing memory easier.
What is padding?	Objects are not a power of two need to be padded (space in memory to fill a block) Padding is wasted space with no data that can't be used.
What is reference counting?	Values are associated with a count of how many references they have. Every value associated with it. A count of how many references it has. Tracking reference. It if has 0 references, then it can be freed.
What happens if a reference counting mechanism does not track and break references cycles?	Chains of values that reference each other are locked and will never be deallocated. Ex. 2 objects reference each other and the memory they take up cannot be allocated because their reference counts will never be 0.

Optional Questions: **What is non-invasive reference counting?** **What is a lifetime?**	Reference counting – a wrapper around data, data is never modified. Rust has "RC" and also "ARC" RC is done dynamically. A lifetime of a variable is how long it is defined. Similar to scope.
(Quiz #9)	
What is a *root set*?	Collections of things that can reference into the store. All the data that is directly accessible to the program.
What does it mean for a variable to *live*?	The variable is useable/in scope. It has a lifetime, it has a future, it hasn't been collected yet.
What is the difference between *truth* and *provability* computationally?	Truth is precisely collecting all garbage and nothing more or less. Cannot be obtained on a turning-complete language. Provability is proving that a variable lives and getting rid of those that are not living. There are things that cannot be computed. Ex. You can't write the perfect garbage collector. Truth: Provability: Distinction between what the truth is and what you can prove. Ex. The halting program. Truth: the ideal of what we can to computer and provability is what we can actually computer. You can substitute computability with provability. Truth would be perfect garbage collection (GC at the right time and place) Not possible in turning-complete machine. Provability: algorithms that attempt to reach truth.
What do completeness and soundness mean in the context of garbage collection?	To be sound means to not accidentally remove anything. Only way to be certain is not by removing anything. TO be complete is to remove everything.
Why is sound, efficient garbage collection so difficult in C and C++?	Change references to values and back to references. Can possibly change non-referenced values. Any value can be a reference. C/C++ allow for references to poof into existence. Anything can be a reference, difficult to do GC because of this.
What is conservative garbage collection?	Makes the assumption that many store locations are not roots and deduces what must be a reference and can be ignored. Results in reasonably effective garbage collection. Conservative GC assumes not everything is a root. Can decide what can be collected or what can be ignored.
Optional Questions: **What is mark-sweep garbage collection?** **What is generational garbage collection?**	Mark sweep is made up of two phases of garbage collection: • Mark anything collectible • Remove anything that is marked. Generational garbage collection is based on the way we write programs. Short lived: many variables. Long lived: few variables. Splits on short and long-lived variables.
Chapter 12 (Quiz #10)	
What is "representation"?	Semantics which span from how we represent data. Representation is how data is structured and the semantics that results from it.
What happens when a language's features are mapped directly to equivalent host language features?	They get the same semantics of the host language. The features get the semantics from the host language.
What are the pros and cons of using existing host language features to represent your language features?	Pros: easy to pass through features, can build on it/provide a starting point, avoids redundancy. Cons: errors from the host language might leak though, programs that shouldn't run can run, need to consider features and semantics and their interactions.
What are two ways the book gives to represent closures?	Closures: • Manual representation (argument, body, environment) • direct mapping (racket lambda/racket environment).
What are two ways the book gives to represent the environment?	Environment: • map: map name to value/location • function: pass name, return value of error.
Chapter 13 (Quiz #11)	
What is a macro?	A macro is a simple form of expression re-writing/a function that rewrites expressions. Macros are forms of syntax rewriting/desugaring.
What is macro expansion?	A form of de-sugaring, the output of de-sugaring can be smaller than the input but usually its larger.
What is the type of a macro? (Put another way, a macro is a function from what to what?)	A macro is a function from (one kind of) syntax) to another kind of syntax (syntax to syntax).
What is #' in a Racket program?	Syntax constructor. Creates a syntax object.
What are guards, and why are they useful?	Predicates that decide whether to follow pattern on a macro. Allow conditional pattern matching. Guards are used in pattern matching "only match this pattern when only some additional condition is met". Used fairly heavily in macros. Guards are predicates on patterns that you match on. (pattern matching operates on structure) Predicates that evaluates macros. Useful because they can stop macro if there is an error (return false).
Why should you not copy code?	Copying code could result in multiple executions of that code.

What is macro hygiene, and why is it good?	Macro hygiene ensures macro expression does not cause previously free variables to be bound. Macro hygiene is renaming all variables with unique names. It is good because it prevents weird behavior.
Optional Questions: **1. What is referential transparency?** **2. What is free variable injection?**	1. Referential transparency: a function can be replaced with its body. The inside of the function is not observable from the world. 2. Free variable injection is adding free variables.
Chapter 14 (Quiz #12)	
What does it mean that the HTTP protocol is "stateless"? (not included in study guide)	It does not store state on the server associated with intermediate computations. State must be maintained elsewhere. Every request is complete, separate, this is no state brought over with each request. A state is not stored on the serve (no associated state for computations). There is no state passed for each request.
What are continuations?	Functions that call other functions. Could eventually return/finish. (not divergent type) A function that tells you what to do next.
What is continuation-passing style?	Converting everything with a series of functions and never returning. A structure to writing all code with continuations.
What is the difference between static and dynamic continuations?	Static: continuation at closure creation. Continuation is determined without running the program. Dynamic: continuation at closure invocation. Determined at runtime.
Quiz #13	
What is a generator?	A generator is like a procedure except it resumes to where it left off.
What is the relationship between continuations and the program stack?	Both are structures for what will run next. Continuations and stack are both structures which keep track of what needs to be done.
What are tail calls?	Recursive function calls at the end of a function. Reusing the same stack frame instead of making a new stack frame. Tail calls are recursive functions at the end of a function. They are used so that only one stack frame is used/re-used.
How do continuations relate to exceptions?	Continuations can be used to implement exceptions. Exceptions can be implemented by using continuations.
What are cooperative and preemptive multitasking?	Cooperative: thread system users manually yield control. Preemptive: time/mechanism automatically yields control without user permission.
Chapter 15	
What is static type checking?	Checking types with consistency with each other, follow type rules. Checking declared types before execution.
What is the type environment?	Maps names to types/binds identifiers to types.
Does introducing types change the semantics of a language?	A type system almost always changes the semantics of a language. Yes. Adding types to your language changes what you can express. Types can help with optimization. Adding types or adding types change what is express-able in a language in a way that requires consideration.
What is strong normalization?	A property in which every expression that has a type will terminate computations after a finite number of steps. Languages that are Turing complete do not have strong normalization because turning complete languages can compute infinitely.
How do desugaring/macros work with type checking?	Pattern matching delegates to desuraging. Macros and desuraging rely on type checking. They work together in "lock step"/interconnected.
What does it mean for a type to be invariant across mutation?	A mutation operation cannot change. When we mutate something (change a variable somehow) we say its type cannot change.
What is a type system?	A type system is a language of types, a set of type rules, and algorithms for application (applying rules).
What does it mean for a type system to be sound?	Type system doesn't lie to you. When type checking, the value passed is the same type as the value returned. The types we decided by the type checker will be the same at run time. To be sound: the type system will not lie to you about somethings type.
What are progress and preservation?	Progress: once something is type checked, it can continue to evaluation. Preservation: The result of this step will have the same type as the original/the type of something will not change after, for example, an expression is evaluated. Progress and preservation show soundness.
Chapter 15	
What are type variables?	Type identifiers. Type variables in types such as the T in C++: vec<T>
What is parametric polymorphism?	Where we parameterize types. Ex: templates in C++. Polymorphism based on types. Function is polymorphic on input types.
What is predicative polymorphism?	Can only pass monotypes. Type parameters can only be filled by monotypes.
What is impredicative polymorphism?	Can pass polytypes or monotypes. They are treated as the same. Type parameters can be filled by monotypes and polytypes.
What is relational parametricity?	Relational functions have limits to what they can inspect. Cannot look at actual values but can do things like delete them or rearrange them.
What is type inference?	Type system can deduce the type of identifiers without the type being explicitly stated.
What is constraint generation and what is it for?	Traverses through code and generates constraints for type checking.
What is unification?	The process used to solve constraints. 2nd step of type checking.
What does it mean for a program's types to be under-constrained and over-constrained?	To be under-constrained means that there is not enough information to make a definitive statement. To be over constrained means that there is conflicting or clashing information about types.

What are the principal types of an expression?	The most general type.
What is let-polymorphism?	Infer what the type is, making it a general type.
Chapter 15 (Quiz #17)	
What are tagged union types?	Tagged union type are data structures which can represent multiple types. Tracks variants.
What are untagged union types?	Similar to tagged unions but do not keep track of members being used.
What is soft typing?	Combines static and dynamic typing.
What does it mean for a type system to be nominal?	Determines whether something is equivalent based on explicit declarations (the name!)
What does it mean for a type system to be structural?	Determining the equivalence of something based on its structure/definition.
What are intersection types?	Types which belong to more than one type.
Why do recursive type signatures require a special constructor?	
What is subtyping?	When one type fits into another. In the context of OO: parent and child class. Subtyping establishes a relationship between two types and we say the types can be substituted for each other. Can pass a student anywhere a person is expected. Two major ways of to think of subtyping. Width and depth:
What is width subtyping?	subtype generally has more fields (more defined, less ambiguity) if we pass a student to where a person is expected, the student fields are dropped.
What is depth subtyping?	Depth checks the subtype relationship between equivalent fields. Are their fields subtypes of each other?
Lecture and Lab Questions	
What are higher-order functions?	Functions that take functions as parameters.
What is partial application?	Partial application function with multiple parameters. Can be given parameters piece by piece and will compute correctly.
What is currying?	Function with n inputs can be converted to n functions with 1 input. They're nested.
What is a lexer?	Function which takes input and process an ordered collection of tokens.
What is a parser?	Takes tokens (lexed values) and makes a parse tree.
What is an interpreter?	An interpreter is a computer program that directly executes, i.e. performs, instructions written in a programming or scripting language, without previously compiling them into a machine language program.
What is ahead of time compilation?	What we're usually familiar with. There's a very clear separation between compile time and run time. Compilation and execution are completely separate. No interaction between two steps. Doesn't know anything about runtime.
Interpretation:	No distinction between runtime/compilation.
What is just in time compilation?	Mix of ahead of time and interpretation. Complication happens at runtime. Only compiles things that actually get used. Keeps track of how many times functions are called. At the beginning of a J-I-T compilation, the system can run a little slow. Has runtime information and can guide compilation.
What is syntax?	Syntax is how a program is written.
What are semantics?	
What is de-sugaring?	
What is pattern matching? Is it based on name or position?	
What is an "unbound" term?	We don't have a definition for a variable. Haskell is lazy and doesn't care if something is unbound only when its being used. Do not have substitution
What is a "bound" term?	\x -> x + y x is bound and y is not bound. "has been defined" in the context of parameters has a declaration somewhere have definition.
What is substitution?	
What are static and dynamic scope?	Static Scope: when values are defined are based Dynamic Scope : what values are defined are based on the
What are deep and shallow binding?	Deep binding: traverse stack if an identifier's value exists. Program stack tracks values of identifiers. Shallow binding: interpreter manually tracks values of identifiers. Deep binding and shallow binding are ways to implement dynamic scoping.
What is the environment?	An environment maps identifiers to values
Can a variable be both bound and free in the same program?	Yes. A variable can be bound and free, depends on scope
What are type classes?	A type class is a type system construct that supports ad hoc polymorphism. This is achieved by adding constraints to type variables in parametrically polymorphic types. (SEE LAB 2)
What is a functor?	
What is an applicative?	
What is a monad?	
What are lifetimes?	A lifetime of a variable is how long it is defined. Similar to scope.
What is concurrency?	You can have more than one thread of control. More than one separate control flow. Does not make statement of when they're going to run.
What is parallelism?	If you have parallelism, you have concurrency. You run multiple threads running parallel to each other.
What is a data race?	Multiple threads have non-synchronized write access to shared data. Data races happen when multiple threads modify something without coordination. Final value of shared data can depend on the order.
What is livelock?	Processes are running but not making any progress. Execution stops.
What is deadlock?	Processes stop. Execution halts.
What is starvation?	One method that can lead to deadlock. Resources do not have access to resources.

What is a race condition?	Anything about order of execution can mess with program correctness. Things outside of the control of the program can break the program.
What are processes?	A process has a thread with resources. Process id. Info about who started it, when, permissions, thread of control. Which can spawn more processes. When a process ends, it takes all of its threads.
What are threads?	A thread of control.
What is a mutex?	Container for mutual exclusion. Allows threads to lock and modify data so no other threads can modify. When they're done, they unlock the data.
Semaphore	Mutex with a counter.
What is safety? (be able to provide multiple examples of something that is unsafe)	Memory safety: access to memory are safely controlled. Ex. Garbage collection can provide memory safety. thread safety: the logic of your program and the correctness of your program should not change if the order of your program changes. Ex. Rust does not have deadlock. type safety: the type system won't lie. When the type checker makes its determinations about the types, those types will be the same at runtime.
Can unsafety cause security problems?	Yes.
What are abstract data types?	An abstract data type is a model of a certain kind of data structure e.g. a Stack. A Stack has push() and pop() operations and that have well-defined behavior. The abstract data type (ADT) itself refers to this model, not any particular implementation in any particular programming language or paradigm
What is the SOLID principle? What do each of the five parts mean?	SOLID is an object-oriented design principle: **S**ingle Responsibility Principle – every class should have on responsibility. **O**pen/Closed Principle – classes should be open for extension (subtyping, inheritance), closed for modification (changing source code). Prevents breaking code. **L**iskov Substitution Principle – You should be able to substitute a child class in the place of a parent class without breaking the code. **I**nterface Segregation Principle – classes should implement interfaces with functions that they won't use. Classes should only implement interfaces with functions they will need. Break up interfaces. **D**ependency Inversion Principle – You want to program against interfaces, not concrete implementations. Being generic. Instead of taking a car, say taking a vehicle.
What is backtracking?	When a goal fails, the program stops and goes back to the most recent spot it can continue at. We can think of backtracking as a tree of decisions. Going back up the tree when something fails (Context of prolog)
What is a programming paradigm? (be able to give examples)	A model for thinking about computation. Core programming paradigms: functional -> functions, no state. It's a particular model for thinking about computation, that's how we think about it, object -> objects, subtyping, relationship between objects. Logic -> think of programming in the context of logic. It's a mental model.

Notes:
Hygienic macros do not capture variables.
Macros and C literally copy and paste.
In C you can do

```
	#define if while
```