348 MCQs - Quiz Questions and Answers

Principles of Programming Languages (Concordia University)

Scan to open on Studocu

QUIZ 1:

1. Which of the following is NOT considered to be an imperative language?

a. Prolog
b. Smalltalk
c. Python
d. COBOL
e. None of the other answers is correct

The correct answer is: Prolog. Imperative languages are those that consist primarily of variable assignments, iteration, and basic control flow manipulation. Most common languages are imperative. In the list of answers, the only obvious exception is Prolog, which is declarative in nature. In this case, language statements indicate what we would like to find as a result, but they don't explicitly tell us how to get there.

2. Static linking refers to the process by which application libraries are loaded at runtime

True
False

The correct answer is False
If libraries are loaded at runtime, we refer to this as dynamic linking. In contrast, static linking occurs before runtime, and all components are combined into a single image.

3. Proper standardization is a property of which language evaluation criteria.

a. Readability
b. Writability
c. Well-definedness
d. Expressivity
e. Portability

The correct answer is: C.

4. The language most commonly used for software such as compilers and operating systems is...

a. None of the other answers is correct
b. Java
c. Python
d. Smalltalk
e. Fortran

C is by far the most common "systems" language

5. The fundamental computer architecture that influenced early language design was known as...

a. None of the other answers is correct
b. The Pipeline model
c. The Variable model
d. The ALGOL model
e. The Von Trapp model

The correct answer is: It's the Von Neumann model, named after the mathematician who promoted the design.

6. Languages are either fully orthogonal or not orthogonal at all. They can not be somewhere in the middle

True
False

No. They usually fall on a spectrum of more or less orthogonal.

7. When discussing the CPU, we indicated that it houses the following primary components...

a. ALU
b. The Control Unit
c. I/O devices
d. Compiler
e. Main Memory

The correct answer: ALC and The control unit

8. Expressivity is a property of readability

True
False

The correct answer: Is false: greater expressivity makes code easier to write concisely.

9. The move from data-oriented design to process-oriented design was a key feature of the impact of programming methodologies on language design.

True
False

The correct answer is 'False': the movement was actually in the other direction - from process-oriented design to data-oriented design.

10. Which of the following statements are false?

a. With a compiler, source code is broken into tokens during the semantic analysis phase
b. A compiler uses a symbol table to keep track of variables and their types
c. Lexical analysis happens before syntax analysis
d. During compilation, compilers optimize intermediate code during the synatx analysis phase
e. Compilers accept bytecode as input

The correct false answers are: a,d,e
With a compiler, source code is broken into tokens during the semantic analysis phase,
During compilation, compilers optimize intermediate code during the synatx analysis phase,
Compilers accept bytecode as input

11. Which of the following languages were associated with the earliest programming domains?

a. C
b. Algol
c. LISP
d. Fortran
e. COBOL

The correct answers are: c,d,e
LISP
Fortran for used for scientific programming
COBOL was used for business programming

12. The general processing logic of a CPU controls something called the Fetch-Decode-Execute loop

True
False

Yes, true, the core objective of the CPU is to fetch the next instruction, decode its purpose, and then execute the logic of the decoded instruction.

13. Which of the following statements are true, if we are referring to pure interpreted languages?

a. Applications are usually relatively slow
b. Data types can be defined at runtime
c. Applications are easy to port to other architectures
d. They are not common today

All of the above: a,b,c,d

14. The process of converting bytecode into native machine code at runtime is known as...

a. API
b. AOP
c. Hybrid
d. JIT
e. RAM

The correct answer is: JIT (or Just In Time) compilation

15. The process of translating textual 'tokens' in the source code into trees that represent the structure of the program is know as...

a. Lexical Analysis
b. Semantic analysis
c. None of the other answers are correct
d. Stemming
e. Binary code generation

The correct answer is: C. None of the other answers are correct

16. Which of the following statements is true?

a. Compilation is an implementation technique typically associated with smaller projects
b. Virtual machines are typically associated with programs that have been converted into bytecode
c. Source code is directly executable in hybrid implementations
d. C is an example of a compiled language, while Java is an example of pure interpreted language

The correct answer is: b. Virtual machines are typically associated with programs that have been converted into bytecode

17. Languages properties that improve reliability include...

a. Full support for variable aliasing
b. Compilation Overhead
c. Generality
d. Type checking
e. Rich support for exception handling

The correct answers are: d and e
Type checking - compilers can find type mismatches early in the development process
Rich support for exception handling - this tends to make code cleaner and easier to maintain

18. The four primary language evaluation criteria discussed in class are...

a. Code re-use, writability, expressivity, and simplicity
b. None of the other answers is correct
c. Cost, compilability, code re-use, and readability
d. Readability, writability, orthogonality, and cost
e. Reliability, Readability, Cost, and Writability

The correct answer is e: Reliability, Readability, Cost, and Writability

19. In terms of language design, which of the following languages would have been impacted by knowledge of the physical CPU architecture?

a. LISP
b. Fortran
c. Prolog
d. C++
e. C

The correct answers: b, d, e. In general, knowledge of the CPU architecture lead designers to map high level languages - with variables, assignment statements, and loops - to the hardware available to them (memory, control unit, registers, etc). So imperative language like C, C++, and Fortran would be in this group.
But languages like Prolog (declarative) and LISP (functional) are quite abstract and have no real connection to any hardware components.

20. The process of identifying the various program components and moving them into memory is called linking.

True
False

The correct answer is False. This process is generally referred to as loading. In contrast, linking refers to the process of preparing references to variables and methods across all units so that the code functions as if it had been compiled as a single unit.

21. In terms of language design, reliability and cost of execution can be hard to optimize at the same time

True
False

The correct answer is 'True': there tends to be a trade-off between the two. For example, run-time index checking typically increases the execution cost of a running program.

22. A preprocessor is used in which of the following languages?

a. Python
b. Javascript
c. Java
d. None of these languages uses a preprocessor
e. Clojure

The correct answer is d. In general, preprocessors are only associated with older, compiled languages such as C/C++.

23. Which of the following are preprocessor directives in C

a. #delete
b. #define
c. #then
d. #macro
e. #import

The correct answer is b.

24. Which of the following components is NOT found in a hybrid language implementation?

a. Syntax analyzer
b. Actually, none of these are part of a hybrid implementation
c. Binary code generator
d. Intermediate code generator
e. Lexical analyzer

The correct answer is: c
Binary code generator - this is not included since we are passing the byte to an interpreter

25. Which of the following is an example of a functional language?

a. Clojure
b. Clojure and LISP are functional, but not Haskell
c. All three of these languages are functional
d. Haskell
e. LISP

The correct answer is c: All three languages - LISP, Clojure, and Haskell - are functional in design. Clojure is a modern form of LISP, while Haskell is a sophisticated functional language with a somewhat different syntax.

26. Hybrid languages implementations would include:

a. Python and Java
b. Java and C++
c. Clojure and C++
d. C and Fortran
e. COBOL and Fortran

The correct answer is a.

QUIZ 2:

1. C++ and Java are two popular Object Oriented languages. Both are direct descendants of which earlier language?

a. Actually, Java and C++ are not connected to any common ancestor
b. APL
c. Simula - Yes, Simula was an early pioneer of data-oriented language design
d. Smalltalk
e. Algol

The correct answer is: Simula.

2. The Lambda Calculus provided the formal foundation of which programming language?

a. Algol
b. Smalltalk
c. Fortran
d. LISP
e. Simula

The correct answer is LISP

3. With respect to Fortran, which of the following statements are true?

a. Early versions were interpreted
b. Storage for data elements was fixed at compile time
c. Extensive support for string processing
d. Array handling was directly supported
e. No direct support for loops

The correct answers are:
Storage for data elements was fixed at compile time - the compiler did this statically

Array handling was directly supported

4. Which of the following was not a characteristic of the Algol language?

a. It had an optional Else-IF clause
b. Arrays could have any number of subscripts
c. Compound statements used curly braces: { } - This is false. Algol used Begin/End blocks
d. Subscripts used square brackets: [ ]
e. Names could be any length

The correct answer is: Compound statements used curly braces { }
Algol used Begin/End blocks

5. Which language is generally considered to be the first to support Classes, Objects, and Inheritance?

a. C++
b. Simula
c. Smalltalk
d. Java
e. Algol

The correct answer is: Simula

6. Which of the following languages is a dialect of LISP?

a. Both Clojure and Scheme
b. Both Clojure and Haskell
c. Haskell
d. Scheme
e. Clojure

The correct answer is: Both Clojure and Scheme

7. Java is considered to be a more portable language than C

True
False

The correct answer is: True. While the compilers and language tools themselves are widely available for both languages, C applications can be far more challenging to port across different architectures (OS and CPU).

8. With respect to Algol, which of the following statements are true?

a. It replaced COBOL as the dominant business language
b. It was the first language whose syntax was formally defined by BNF (Backus Naur Form)
c. Virtually all subsequent imperative languages are based on it
d. It was the standard way to publish algorithms for over 20 years
e. It was a failure commercially

The correct answers: b, c, d

9. In which language might you expect to use a statement like the following:
sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y).

a. Python - No, this is logic-based rule we would expect to see in Prolog
b. Smalltalk
c. None of the other answers is correct
d. Prolog
e. APL

The correct answer is: Prolog

10. Which language uses inference to determine the truth of given queries?

a. None of the other answers is correct
b. Simula
c. Erlang
d. APL
e. Smalltalk

The correct answer is: None of the other answers is correct.
Inference is used by Prolog (programming in logic).

11. C is perhaps the most innovative of all languages.

True
False

The correct answer is: False.
This is definitely not true. C is perhaps the most influential of all languages, in that many subsequent languages used elements of its design and syntax. However, it is not particularly innovative, as it didn't really introduce many dramatically new ideas.

12. Most languages are designed by one person, or perhaps a very small team. Which of the following languages was designed by a committee?

a. Python
b. Smalltalk
c. Simula
d. COBOL
e. C

The correct answer is: COBOL
COBOL was designed by committee of manufacturers and Department of Defence (DoD) members

13. Which of the following early languages had the greatest influence on the design of languages that came later?

a. Fortran
b. PL/1
c. COBOL
d. LISP
e. Basic

The correct answer is: LISP
LISP influenced almost all of the functional languages

14. With respect to Fortran, which of the following statements are false?

a. Fortran supported user-defined subprograms
b. The execution speed was very good, within a factor of 2 of machine code
c. There was no IF statement
d. Because it was an early language, the compiler was easy to develop -
e. Few companies actually used Fortran since COBOL was much more useful at the time

The correct answers are: c, d, e
For c. It is almost impossible to do anything useful without an IF statement
For d. They were really hard to develop in the early years
For e. Fortran was very widely used

15. Assembly Language refers to the binary code that corresponds to the machine's basic Instruction Set

True
False

The correct answer is false. This is perhaps a little tricky. But the term machine code refers to the binary instructions, while assembly language/code refers to the one-to-one mapping of human readable mnemonics to the machine instructions.
In other words, machine code is for computers and assembly code is for humans.

16. In which of the following languages would we expect to see a line like the following: MULTIPLY Num1 BY Num2 GIVING Result

a. Algol
b. COBOL
c. Fortran
d. APL
e. Simula

The correct answer is: COBOL was designed to look almost like written text

17. Sophisticated OOP concepts like dynamic binding and reflection were introduced by which language?

a. Smalltalk
b. Java
c. Simula
d. C++
e. None of the other answers is correct

The correct answer is: Smalltalk is generally considered to be the first complete OO language

18. Which of the following statements are false?

a. In Smalltalk everything is an object, including ints and floats
b. Java influenced the design of C++ - No, it was the other way around
c. Simula is closely associated with the design of graphical desktops - No, that was Smalltalk
d. Both C++ and Java allow non-OOP code to be written - No, C++ does, but Java does not
e. C++ was developed by Bjarne Stroustrop

The correct answers are b, c, d.

19. In LISP, the head of a list is represented by what function?

a. CDR - No, it uses CAR for this purpose (CDR is the rest of the list)
b. None of the other answers is correct
c. LHEAD - No, it uses CAR for this purpose
d. FUN
e. T

The correct answer is b.

20. Assembly languages represent the first high level languages.

True
False

This is false. Assembly language represents the one-to-one mapping of human readable mnemonics to machine instructions. As such, there is no abstraction to more intuitive programming constructs at all.
True high-level languages begin with Fortran, COBOL, and LISP.

21. The phrase "write once then throw away" is often associated with which language?

a. Smalltalk
b. APL
c. Java
d. Prolog
e. C

The correct answer is: APL code is very hard to read

22. Ruby and Lua are modern languages that would be most similar to which of the following languages?

a. Java
b. Objective-C
c. C
d. Python
e. C++

The correct answer is Python - Yes, even though the syntax can be quite different, each is a flexible, dynamic language

23. Which of the following languages was originally designed with only two data types?

a. Smalltalk
b. LISP
c. Simula
d. Algol
e. C

The correct answer is: LISP, which only had atoms and lists

24. Fortran was/is widely used but had little influence on the syntax of other language that were designed in subsequent years

True
False

This is true. Fortran's impact was really just that it worked, and it therefore encouraged other people to keep experimenting with new ideas. The syntax itself, however, didn't really have much of an impact on newer languages.

25. The first major, fully dynamic language was...

a. Java
b. Algol
c. Simula
d. COBOL
e. APL

The correct answer is e. - Yes, almost everything happened at runtime

QUIZ 3:

1. Which of the following statement is false?

a. When we include a header file, the source code from the header file is inserted directly into the current file
b. In order to call a function from another source file, we must include that function in a header file
c. In C, the size of an integer (i.e., the number of bytes) is fixed in size across all computing platforms.
d. A function prototype does not include the return type
e. None of the above are true

The correct answer is: E, all of these statements are false (for B, we “should” use headers, but we don’t actually have to)

2. In the code below, we can say the following:

a. The value of count is changed to 43 in both foo and the calling function
b. The value of count is changed to 43, but only in the foo function
c. The value of count is changed to 43, but only in the calling function
d. This value of count is changed to 43 in foo and 44 in the calling function
e. This would produce a syntax error

```
int foo(int count){
    count = 43;
    return count + 1;
}
```

The correct answer is: B, the local value is changed to 43, but this doesn't impact anything else

3. Assume that we are compiling C code in a machine that uses 4 byte integers. For the line listed below, how many bytes of storage are allocated by malloc()

a. 10, one for the pointer to each integer
b. 4, for the 4 required for the intBuf pointer
c. 0, since malloc use the stack, which uses automatic allocation
d. 40, 4 bytes for each integer
e. 80, one for each pointer, and one for each integer

```
int *intBuf = (int *)malloc(10 * sizeof(int) );
```

The correct answer is: D, 40 bytes, 4 bytes for the space required for each integer.

4. We are working on a computer that has a 64-bit word size, and ints are 4 bytes each. So let's now say that we are importing a header file. When the C compiler reads this header file, how many bytes will it allocate when it encounters the following?

```
struct foo {
  int x;
  int y;
  int z;
};
```

a. 16 - No allocation is made for the declaration
b. 24
c. 4
d. 12
e. 0

This is a bit of a trick question (I wouldn't do this on a test, but it's OK for a self-test quiz)
So you might look at this and say the following:
1] an int is 4 bytes in size
2] The word size is 64 bits, which is 8 bytes
3] There are three ints in the struct, so this is 12 bytes
4] BUT, structs are "padded" so that they end on an even word size, so the total size of the struct would be 16 bytes
This is really good logic, except for one thing. Header declarations are just a template for the struct, they do not actually create anything. So NO bytes are allocated when the header is read. This would not happen until we actually created a struct of this type, as in:

struct foo myFoo;

The correct answer is: 0

5. Which of the following statements are false. You may select more than one.

a. We can have multiple versions of a given function in C, but only if each version has a different number of parameters.
b. If an array is passed to a function, we can use the sizeof operator inside the function to determine the number of elements in the array
c. Good C design dictates that we include relevant .h header files in the current .c source files but that we should NEVER include source files in header files
d. With C, it's possible to have static and non-static functions in the same file.
e. The gcc option -Wall will display warnings but it will NOT automatically abort the compilation process.

The correct answers are: A, B
A is false. C has NO method overloading at all. We can have multiple versions of a given function in C, but only if each version has a different number of parameters.
B is false. When sizeof is used on function parameters, it will only return the size of the variable itself (in this case, the size of the pointer). If an array is passed to a function, we can use the sizeof operator inside the function to determine the number of elements in the array
C is true. Do NOT include .c source files in .h header files. This is almost always a terrible idea, even if it compiles and seems to run. We add prototypes, declarations, etc, to included files, NEVER implementation code.
D is true. The static functions are private to the file (e.g., "helper" functions), while the non-static functions represent the public API.
E is true. -Wall will detect and display many types of warnings, but we would have to use -Werror (or something similar) to tell the compiler to stop if it encountered any type of warning.

6. Consider a small C application that has ten functions in total, all of which will be executed once as the program runs. Without knowing anything else about the source code, what is the minimum and maximum number of stack frames that could in theory be active at any one time?

a. min=1, max = 1
b. min = 2, max = 10
c. Min = 1, max = 10
d. Min = 0, max = 10
e. min = 10, max = 10

There are a couple of things to keep in mind:
1] A program has to have a main() function, which is always active
2] When another function is run, there would be at least 2 frames on the stack (main plus function 2). However, main can do some of its own computation before/after calling the other functions. So there would be just one frame at those times.
3] If main calls function #2, and 2 calls 3, and 3 calls 4, etc, then there would be up to 10 frames active in that case.

The correct answer is: C, the minimum is 1 and the max is 10.

7. Which one of the following is NOT one of the standard C header files?

a. list.h
b. stdlib.h
c. stdio.h
d. Actually all of these are standard C header files
e. string.h

The correct answer is: A. Even if you do not remember most of the headers, hopefully you recall that C has no list data type, so it would clearly not have a header file to define list functions.

8. In C, a string must be terminated with \n in order for the string libraries to identify the end of the string.
Select one:

True
False

The correct answer is: False. In C, we terminate strings with the NULL character, \0.
In contrast, \n is just a newline character, which is a perfectly valid character in a string. So something like "I\nam\nhappy\ntoday" would print

```
I
am
happy
today
```

assuming a \0 had been properly appended to the string.

9. Assume that we are compiling C code in a machine that uses 4 byte integers. For the line listed below, how many bytes of storage are allocated by malloc()?

```
int *intBuf = (int *)malloc(10 * sizeof(int) );
```

a. 10, one for the pointer to each integer
b. 40, 4 bytes for each integer
c. 80, one for each pointer, and one for each integer
d. 4, for the 4 required for the intBuf pointer
e. 0, since malloc use the stack, which uses automatic allocation

The correct answer is: B. The total will be 40 bytes: 4 bytes for the space required for each integer

10. What is the purpose of the second statement in the following C code?

```
time_t  t;
srand((unsigned) time(&t));
```

a. It ensures that negative numbers will not be generated
b. It ensures that the random numbers will fall between 0 and the value of the current time
c. It ensures that each sequence of random numbers will be unique
d. The random numbers will be time values
e. It ensures that the next random number will be assigned to the variable t

The correct answer is: C, "srand" seeds the random number generator with the current time so that the subsequent numbers will have a unique starting point.

A is false. "srand" seeds the random number generator with the current time so that the subsequent numbers will have a unique starting point.

11. Which of the following are valid statements in C (or all of them)?

a. char baz[3] = {‘1’, ‘2’, ‘\0’};
b. char* = "";
c. Actually, they are all valid
d. char *fluffy = “fluffy”;
e. char* bat = NULL;

The correct answer is: C. Actually, they are all valid
B is true. It is just an empty string
E is true. It's an uninitialized string pointer

12. Which of the following statements is true?

a. malloc utilizes the stack for memory allocation
b. C pointers refer to physical memory addresses
c. It's possible for malloc to return a NULL pointer
d. by default, malloc returns a pointer to an int
e. malloc initializes new memory, while calloc does not

The correct answer is: C
A is false, it uses the heap.
B is false, they refer to logical addresses.

13. It is not possible for the memory associated with a statically allocated variable to exist for a shorter period of time than either a dynamically or automatically allocated variable
Select one:

True
False

The correct answer is True. Statically allocated variables are those that are known at compile time. As a result, space for them is set aside in the compiled program image, even before it runs. Memory for these variables will exist for the entire lifetime of the running program.
The space for dynamically and automatically allocated variables is determined while the program is running. In the most extreme case, they could be associated with the main() method, which exists for the entire lifetime of the program. But, even so, they can not be created before (or cleaned up after) static variables. Moreover, in the vast majority of cases, their lifetime is much shorter than that of static variables.

14. Which of the following statements is true?

a. Neither can be easily ported to other platforms without considerable effort.
b. In general, C applications are considered to be quite portable relative to Java applications.
c. Both can be ported easily without requiring any additional testing.
d. In general, Java applications are considered to be quite portable relative to C applications.

The correct answer is: D. While it is possible to port applications from either language to other platforms and architectures, it is generally much easier to do this with Java.
This is the case because the Java language is designed for an artificial environment (the Java Virtual machine). As such, its data types and library functions are standardized, allowing the underlying virtual machine to handle the architectural differences so that the programmer does not have to worry about this. In contrast, C is a low-level compiled language and, as such, source code is translated directly into the format relevant to each individual machine. So while the C compilers themselves have been ported to almost even possible environment, the programmer must still ensure that the compiled program runs properly on each target platform.

15. Which of the following is a valid array declaration in C?

a. X[10] int;
b. int array[10];
c. Actually, none of these will work.
d. bar int(5);
e. foo int[5];

The correct answer is: B.

16. It is generally suggested that the fprintf function be used to display errors within C programs. Why?

a. fprintf is implemented in assembly language
b. Actually, there is no such function as fprintf
c. fprintf can write to files as well as the console - No, they both work the same way in terms of output targets
d. fprintf is a part of the standard C libraries
e. fprintf is an unbuffered IO stream

The correct answer is: E. It is certainly true that one can use a simple printf to display error messages. However, the problem with printf is that, for reasons of efficiency, its output is buffered. In other words, it is held in a buffer until enough text has accumulated in order to justify the overhead of writing to the console. For general I/O this makes sense. However, for error display, we do not want critical messages to get stuck in a buffer for long periods of time. Consequently, fprintf provides an unbuffered error stream that sends its output to the console (or at least to the Operating System that controls the console) as soon as the function is invoked.

17. Which of the following statements are true. You can select more than one.

a. C will ignore duplicate calls to free()
b. In C, & is the address operator
c. The largest block of addresses in a program space is associated with the heap
d. C uses reference counting for garbage collection
e. If you do not use dynamic memory allocation, you do not need to cal free()

The correct answers are: B, E.
A is false, it will abort since you can not free an address that has already been released.
B is true and returns the address of the variable that it is associated. In contrast, * indicates that something is a pointer type - it does not provide the address itself.
C is false. This might be true, but probably not. In practice, most addresses in the process space are currently unused, since most programs only needs modest amounts of memory for the stack, heap, etc.
D is false. Python does this but definitely not C, which has no native garbage collection at all.
E is true. free() is used to clean up memory allocated dynamically with malloc, calloc, or realloc. If you only use static or automatic memory allocation, there is no reason to call free.

18. Which of the following statements is false? (If you think they are all false, there is an option for that too).

a. In order to call a function from another source file, we must include that function in a header file.
b. The #include directive is part of the C language
c. A function prototype does not include the return type
d. Actually, ALL of the other statements are false
e. In C, the size of an integer (i.e., the number of bytes) is fixed in size across all computing platforms.

The correct answer is: D.
C is false. Since C is a low-level language and integer size is depends on the CPU that is being used on the current computer. But all of these statements are actually false.

19. We want to include one of our own header files in the current C source file. However, the header file is in another folder. To allow the compiler to find the header file, we must add the other folder to the current CLASSPATH environment variable.
Select one:

True
False

The correct answer is False. C has no CLASSPATH variable, as we would see with languages such as Java or Clojure. Instead, you would have to take another approach, such as passing the folder/path to the compiler as a command line option.

20. In C, the following two declarations are equivalent

```
char* argv[ ]
char** argv
```

Select one:

True
False

The correct answer is: True. Both refer to a pointer to an array of one of more pointers.
In C, there is no built-in "array" data type, even though we use [ ] to represent an array of values. Instead, an array is just a reference to a memory location where some values are located. In the case of strings, this would be a sequence of character pointers. In other words, each individual character pointer would reference a specific string, as in:

[0] -> "dog"
[1] -> "cat"
[2] -> "bird"
In this case, argv is an array of string pointers. So we can define it as:
i] a pointer to a pointer to a char (char** argv); OR
ii] a pointer to an array of chars (char* argv[ ])
These two declarations are equivalent.

21. In terms of function visibility in C, which of the following is true?

a. By default all functions are globally visible throughout the application and we use the extern modifier to make them private to the file
b. C functions are always global and there is no way to make them private to the file.
c. By default all functions are private and we use the static modifier to make them global to the application
d. By default all functions are private and we use the extern modifier to make them global to the application
e. By default all functions are globally visible throughout the application and we use the static modifier to make them private to the file

The correct answer is: E.

22. Which of the following statements is true?

a. If main is not the first function, we MUST include a prototype for it at the top of the file
b. the main() function can be placed anywhere in the current source file
c. main() MUST be the last function in the source file
d. main() MUST be the first function listed in the source file
e. If we don't include a main() function, the compiler will add a default version of main()

The correct answer is: B
A is false. We don't need a prototype for main(), since no other function is going to invoke it.

QUIZ 4:

1. Which of the following statements is true?

a. In Python, queues are efficiently implemented using lists, but stacks are not as efficient
b. In Python, both stacks and queues are efficiently implemented using lists
c. In Python, neither stacks nor queues are efficiently implemented using lists
d. In Python, stacks are efficiently implemented using lists, but queues are not as efficient

The simple answer is that stacks work a little better with basic lists. This is simply because lists add elements at the end of the underlying array. For stacks, this is fine, since values are pushed and popped at the end of the array.
For queues, however, we would have to pop at the head of the array, which would cause expensive re-organization.

2. In Python, how would a subclass Bat call the constructor of its parent Dog?

a. Bat.Dog()
b. Dog.Bat()
c. Dog.__init__(self)
d. __init__(parent)

e. __init__(self, Dog)

The correct answer is c.

3. In Python, what is a deque?

a. A set data structure - No, it's a double-ended queue
b. An IO channel for error reporting
c. A depth equalized unary exception
d. A binary tree data structure
e. A data structure that supports stack and queue operations

The correct answer is e.

4. What would the following Python code produce

```
print(r"G:\my\noo")
```

a. It would display "G:\my\noo" - Yes, this is a raw string that would print the content without interpreting escape chars
b. it would print "rG:\my\noo"
c. It would print
"G:\my
oo"
d. It would produce an error since \m is not a valid escape character
e. It would produce an error since the leading 'r' is outside the string
The leading 'r' indicates a raw string. This means that escape chars are ignored, so the associated string would print exactly as it is written.

The correct answer is a.

5. Recent Python versions are in the 3.x line. However, like all programming languages, Python 3.x is backwards compatible with the earlier Python 2.x, since otherwise we would not be able to run existing 2.x programs with the new Python interpreters.

Select one:
True
False
Perhaps somewhat surprisingly, this is FALSE. The changes introduced in the version 3.x line of the language were so significant that backwards compatibility could not be supported. So while 3.x programs are often very similar to 2.x programs, there are differences in syntax/structure that prevent the new interpreters from running ALL older programs.

6. Assume that we have a Python module called jump.py. This module contains the functions calc(), tax(), and sell(). In the current source file, we would like to be able invoke these functions without specifying the full path to each [e.g., we can invoke sell() instead of jump.sell()]. Which of the following statements would accomplish this?

a. from jump import *
b. import jump.*
c. import all from jump
d. from jump import all
e. import * from jump

The correct answer is a.

7. In Python, sequence data types include...

a. sets, tuples, and strings
b. Lists, sets, and tuples - sets are not sequences, as they are unordered
c. Lists, strings, and tuples
d. strings, dictionaries, and tuples
e. tuples, dictionaries and strings

The correct answer is c. In short, sequences are composite data structures in which the included elements can be ordered (i.e., they can be referenced by index positions)
In Python, this includes lists, tuples, and strings. The other data structures - dictionaries and sets are not ordered.

8. With respect to Python dictionaries, which of the following is correct (you can select more than one)?

a. A key can be a string but not an int - No, ints are fine as keys
b. There is no fixed order for the keys
c. Keys must be mutable d. The following statement would createa python dictionary with a two key/value entries: - Not quite. It should be:

foo = {“bob” , “Halifax” : “Helen” , “Toronto”}
foo = {“bob” : “Halifax” , “Helen” : “Toronto”}

e. Adding a key that already exists is not an error

The correct answers are b and e(The new entry simply replaces the current value).

9. In terms of Python compilation, which of the following statements is not true?

a. The bytecode files uses a .pyc suffix
b. Python uses just-in-time compilation
c. In Python import directives are not processed by the bytecode compiler
d. The main program module is not compiled
e. Source code is converted into an intermediate byte code format the first time it is encountered

The correct answer are a and b.

10. Let's say that we want to provide a constructor for a new class called Foo. The constructor will be used to create an instance variable called dog and assign the value "spot" to it. Which of the following represents a valid constructor for this purpose?

a.
```
class Foo:
  init(self):
    self.dog = “spot”
```

b.
```
class Foo:
  __init__(self):
    self.dog = “spot”
```

c.
```
class Foo:
  Foo(self):
```

```
          return “spot”
```

```
d.    class Foo:
        dog = "spot"
        __init__(self):
          return self.dog
```

```
e.    class Foo:
        Foo(self):
          self.dog = “spot”
```

The correct answer is b.

11. Which of the following does the Python language NOT support (or perhaps it supports all of them)?

a. object-orientation
b. Actually it supports all of these things
c. static typing
d. white space indentation
e. garbage collection

The correct answer is c - We don't do things like "int x" or "float foo"

12. What would be assigned to the variable x after execution of the following python code?

```
x = "4" * 4
```

a. 16
b. "4444"
c. This would produce a runtime error since python would flag this as an invalid operation - No, this would actually assign "4444" to x
d. 256

The correct answer is b.

13. Which of the following statements would represent a valid representation of class inheritance in Python? You can choose more than one

a. class Dog(Animal):
b. class Dog extends Animal:
c. class Dog: Animal
d. Dog(Animal):
e. class Dog(Animal, House):

The correct answers are a(this is the most basic form of inheritance) and e(this is just multiple inheritance, which Python allows).

14. Does the following code represent valid Python syntax?

```
abc = "abc"
a,b,c = abc
```

Select one:
True
False

Yes, this is TRUE. Note the following:
1] abc is simply a variable that holds the string "abc"
2] A string is a python sequence
3] We unpack the sequence of three characters into three variables
4] This assigns the "a" to a, "b" to b, and "c" to c

15. What is the computational costs of lookups (i.e., searches) in a Python dictionary?

a. O(1)
b. The cost is random
c. O(n)
d. O(n log n)
The correct answer is O(1).

16. Which of the following statements are false (you can select more than one)?

a. The self parameter in a Python object can actually use any name. In other words, self is not a Python keyword
b. We do not need to use the "self" parameter when defining a new class (since it is understood to be there), but we must use it when invoking the object's methods
c. Like Java, Python uses the class keyword to define new classes
d. Like Java, python uses the new() method to create new objects
e. Yoo can not use regular functions and objects in the same Python file. In other words, all of the code must be either object oriented or non object oriented

The false statement: d(Python uses an even simpler syntax [as in "MyFoo = Foo()"]), e(You can mix the two styles together with no problem), b(it's the other way around)
The true statement: a(you should probably use self since it is the standard convention) and c

17. Which of the following list operations is guaranteed to have O(1) cost complexity in python?

a. update a value
b. append a value
c. insert a value - This is O(n) on average since many cells will need to be shifted to make room
d. foo.sort() # for a list named foo
e. foo.index(x) # for a list called foo and a value = x

The correct answer: a

18. Consider the Python code below. Based on this, we can say that the print statement will display...

```
bar = "fire"
global foo = “man”  <-- syntax error
def correct():
```

```
    foo = “woman”
print(bar, foo)
```

a. "fire"
b. This will not work because there is a syntax error
c. "fire woman"
d. "fire man"

The correct answer is b (The global keyword should be used INSIDE the function to indicate that we should use the global version of foo, not a local version)

19. Let's say that we want to create a populate a set and we have the code listed below. What can we conclude?

```
foo = { }
foo.add("a")
foo.add("b")
foo.add("c")
foo.add("a")
```

a. This would produce an error since you can not add two "a" values to a set
b. This would work fine and the set would contain {"a", "b", "c"}
c. This will produce an error since the first line should be foo = set()
d. This would work fine and the set would contain {"a", "b", "c", "a"}
e. This would fail because Python does not have a set data structure

The correct answer is c (Yes, foo = { } would create an empty dictionary and the subsequent add operations would fail)

20. Which of the following statements is true?

a. Python strings are immutable but lists are mutable
b. Python strings are mutable but lists are immutable
c. In Python, both strings and lists are mutable
d. In Python, both strings and lists are immutable

The correct answer is a.

21. What does the following Python statement do?

```
foo = ()
```

a. It creates an empty tuple and assigns it to foo
b. It creates an empty list and assigns it to foo
c. This will produce a syntax error since it makes no sense
d. It assigns an anonymous function to foo
e. Absolutely nothing - it's a "no op" (no operation)

The correct answer is a.

22. Which of the following is NOT one of the locations the Python interpreter will use to locate additional modules?

a. A default system folder containing the Python system libraries
b. The python project.py configuration file
c. The folder in which the application was run
d. The PYTHONPATH environment variable - This is a valid location

The correct answer is b(- Correct. I just made this up)

23. With respect to Python, which of the following statements is true? You can choose more than one).

a. Using an underscore prefix (e.g., _foo) on an object's instance variable makes this variable private (i.e, it can not be modified from outside the object) - No, not quite. This is a convention in Python to indicate that the programmers considers this variable to be private, but it does not actually prevent the variable from being modified
b. Python provides the private and public keywords, but it does not include protected
c. If multiple constructors are provided for a class, they are named __init1__, __init2__, __init3__, etc.
d. New instance variables can be added to an object at runtime
e. All variables in Python are instance variables. There is no notion of a shared class variable.

The correct answer is d.

24. Assuming that we have a variable foo that holds the string "birdhouse", which of the following assignments would be correct (you can select more than one)?

a. foo[2:]   # would be "rdhouse"
b. foo[3:-2]   # would be "dhou"
c. foo[:4]   # would be "bird"
d. foo[4:44]   # would be "house"
e. foo[1:2] #would be "i"

All of these are correct

25. Assume that we have a Python module called name.py that includes the code below. It also imports a module called main.py. We now invoke the application as "python name.py". What would be printed?

```
if __name__ == "__main__":
  print("this one")
else:
  print("that one")
```

a. nothing would be printed since this code would not be executed at all if it is not located in main.py
b. "that one"
c. "this one" and then "that one"
d. "this one"
e. This would be a syntax error

The correct answer is d.

QUIZ 5

1. If we wanted to apply a function to a sequence of values in clojure, such that each value was converted into a new value, we might do so with...

a. conj
b. Reduce
c. filter
d. a closure
e. map

The correct answer is e: map. We are mapping the values to the function to create a new sequence.

2. In Clojure, the logic of the IF and DO forms are combined in...

→ a. when
b. do-if
c. while
d. cond
e. You can't combine IF and DO; this doesn;t really make and sense

3. In Clojure, the command interface is known as the...

a. FORM
b. BEAM
→ c. REPL - Yes, the READ-EVALUATE-PRINT-LOOP
d. CLJ
e. EPFL

4. Which of the following is a valid Clojure map? You may select more than one:

→ a. {"dog" , "spot" , "cat" , "rags"} - Yes, the commas will just be ignored
b. {"dog" : "spot" "cat" : "rags"} - No, we can't use the : symbol
→ c. {"dog" "spot" "cat" "rags"} - Yes, this is the standard method to create 2 key/value pairs
d. {"dog" "spot" : "cat" "rags"}
→ e. {"dog" "spot" , "cat" "rags"}

5. Which of the following statements are true (you may select more than one)?

a. In Clojure, some functions will return NULL - In Clojure there is no NULL value, it is known as nil
b. Clojure uses postfix notation for mathematical expressions - No, it uses prefix notation
c. Clojure runs on top of the Beam virtual machine - No, it runs on the Java VM. erlang uses Beam
→ d. (+ 4 4) is a valid Clojure expression - Yes, this is a simple addition
e. In Clojure, parentheses are only required if a function has at least one parameter - No, they are always required, even if a function takes no parms [e.g., (main)]

6. Which of the following would represent an anonymous Clojure function that doubles the value of the input parameter? You may select more than one option

a. (defn [x] (+ x x))
b. (def [x] (+ x x))
c. ([x] (* 2 x))
d. (anon [x] (* 2 x))
→ e. (fn [x] (+ x x))

7. In Clojure the following two expressions are equivalent:
(def foo (fn [] “flubber”))
(defn foo [] “flubber”)
Select one:
→ True
False

This is true. The defn syntax is just a shortcut for creating an anonymous function and then assigning that function to a label.

8. What does the following block of Clojure code actually do?

```
(defn foo
  [baz moo]
  (baz moo))
```

a. It defines a future - No, that's erlang
→ b. It defines a function that accepts a function and some data as parameters and then applies the passed function to the passed data.
c. It defines a closure - No, it is not returning a function
d. This would just produce a syntax error
e. It creates and returns a new anonymous function

9. With respect to Clojure maps, which of the following statements is false?
→a. It is not possible to create an empty map - This is not true. We can do this simply with { }
b. The get function can be used to retrieve values by key
c. Clojure can use keywords as map keys
d. The get-in function can be used to access nested maps
e. maps can use default return values when the key is not found

10. Which is the following statements about function visibility in Clojure is NOT true?
a. A clojure function can be made private with the following syntax
(defn- flip [moo] (+ moo moo))
b. All functions are public by default
c. There is no protected or public keywords in clojure
d. A clojure function can be made private with the following syntax
(def ^:private flip [moo] (+ moo moo))
→ e. We can use (ns- foo) to make the foo namespace private - No, this is not true, and it wouldn't make much sense to do this since we would not be able to access any functions in foo

11. In Clojure, which of the following would be considered a falsey value value if returned by a function? (you may select more than one)

a. "" ; this is an empty string
b. 0
→ c. false - Yes, this is one of them. The other is nil
d. 1
e. -1

12. Given the Clojure expression below, what would be the return value when the code is executed?

```
(if (> 10 (+ 2 6))
  (do
    (println "good")
    "bad"))
```

→ a. "bad" - Yes, the do block would be executed and it returns "bad" as the result of the final expression
b. "good" - No, it would print "good" but return "bad"
c. This would produce an error since the do block has no return function - No, the return value is the result of the last expression
d. nil
e. This would produce an error since there is no matching else block

13. What is the output of the println expression when the following code is executed? Note that the code works, there are no errors.

```
(defn foo
  [numbers]
  (if (not-empty numbers)
    [ ]
    (conj (foo (rest numbers)) (first numbers) )))

(def numbers [1 2 3 4 5])
(println "foo:" (foo numbers))
```

a. foo:
→ b. foo: [ ]
c. foo: [5 4 3 2 1]
d. foo: [1 2 3 4 5]
e. foo: nil
You will probably notice that this looks a lot like the flip function in the notes. However, there is one small difference. The empty check has been changed to not-empty. This will make the recursion stop on the first iteration and just return an empty list.

14. Which of the following languages support multi-arity functions or methods? You may select more than one option)

→ a. Erlang
→ b. Python
c. C - No, definitely not. No function overloading is allowed.
→ d. Java
→ e. Clojure

Note that arity refers to the number of parameters passed to a function. Multi-arity functions are therefore those that can be overloaded with different sets of parameters.

15. Let's say that we have a vector with 10 elements. Internally, the array is 40 bytes in size. We want to print the values in this vector with recursion, so we will invoke the relevant function 10 times, once for every value in the vector.

What is the minimum numbers of additional bytes that Clojure would use in order to guarantee immutability of the vector across the recursive calls.

a. 40 - No, zero additional storage is required.
b. 10
→ c. 0 - Yes, this is a read-only function so the vector never needs to be copied/updated.
d. 4
e. 400

This is a bit of trick question (I would not ask this on a test). But the simple answer is that if all we need to do is print values, no updates to the array are required. This means that we do not have to add any additional data structures to represent intermediate forms of the array. Therefore 0 additional bytes are required.

16. What does the following Clojure expression represent?
#{12 “dog” “cat”}

a. This is a syntax error - the leading # is an erlang element - No, this is a valid set
b. This is a syntax error because the map does not have a matching set of key/value pairs
→ c. A set - Yes, the leading # is needed to distinguish this from a map
d. A map
e. A record - There are no records in Clojure

17. Which of the following statements about Clojure namespaces and modules is true. You may select more than one option

→ a. The default namespace is called 'user' - Yes, this is what you get by default with the Clojure command line interface
→ b. namespace values are relative to the underlying CLASSPATH
→ c. the namespace path a/b/c would map to (ns a.b.c) in the Clojure source file

d. A namespace can only be defined in one of the source files that makes up the application - No, each of our source file modules will likely define its own namespace, which will represent the functions in that module.
→ e. We do not need header guards when including Clojure modules

18. Which of the following statement are true, with respect to Clojure lists and vectors? You may select more than one option.

→ a. Accessing a value at some arbitrary position is (1) for vectors, but O(n) for lists - This is true. Vectors use array-type indexes so access is immediate, but we must iterate through lists in order to get to the proper position.
b. (def foo (12 "a" "dog")) would create a list with 3 elements and assign it to foo
→ c. A list can contain other lists
→ d. `(+ 2 2) is a valid list - Yes, the leading quote symbol indicates that this is not a function, but a list of three elements
→ e. Vectors and Lists use subscript indexes that start at 0 - This is true for vectors, but lists must use nth to find value

19. If we entered the following at the Clojure command line, what would be the result?
(* (- (+ 3 4) (* 4 1)) 9)

a. -29
b. 8
c. None of the other answers are correct
→ d. 27
e. -6
This would be resolved by computing the "inner" functions and then working outward, as follows:
3 + 4 = 7
4 * 1 = 4
7 - 4 = 3
3 * 9 = 27

20. What would the following expression represent when used in a clojure program?
#(< % 5)

a. we are creating a list by selecting all values less than 5
b. It an anonymous function that with apply the < operator and then the mod operator
→ c. An anonymous function that iteratively substitutes the values of the sequence following this function into the % position
d. It's a literal list of three elements - No, we use the quote character for that
e. This is shorthand for a lazy sequence in clojure that looks at numbers less than 5, but it only generates these numbers as it needs them

21. Which of the following is NOT a valid Clojure expression? (or perhaps all are valid?)
a. (print “dog” "house")

b. (= 1 1)
c. (println “Value:” 12)
d. (not true) - Yes, this just evaluates to false
→ e. Actually all of these expressions are valid

QUIZ 6

1. Which of the following statements are true? You can select more than one option

a. In Erlang, anonymous functions are define with anon - No, Erlang uses fun for this
→ b. In Erlang, the counterpart of Clojure's reduce function is fold
→ c. Erlang has a foreach function
→ d. Erlang modules are defined with the -module attribute
→ e. When searching for modules, Erlang's code path is defined in the ERL_LIBS environment variable

2. The following Erlang code does not compile properly. What is the the basic error?

```
baz() ->
   receive
      {Source, {man, Name}} ->
         send Source {"Received man: " ++ Name};
      {Source {woman, Name}} ->
         send Source  {"Received woman: " ++ Name}
   end.
```

a. Source should be an atom, not a variable.
→ b. Sending to the Source should be written as

```
Source ! {"Received man: " ++ Name};
```

or

```
Source ! {"Received woman: " ++ Name};
```

- Correct. The ! operator is used for the send. There is no actual send() function
c. The receive should not end with end.
d. There should not be a semi-colon at the end of the first receive pattern - No, this is fine, as this just tells us that there are two different versions of the receive function, one for man and one for woman
e. baz() should be called listen()

3. What can we say about the following Erlang receive logic? You can select more than one answer

```
foo() ->
    receive
       {words, Msg} ->
          io:fwrite("Msg in foo: ~s \n\n", [Msg]),
          foo()
    after 3000 -> true
    end.
```

a. This won't work since foo() would have to be called listen() - No, the function that contains receive can be called anything we like

b. It will write the Msg back to the sender's message queue - No, it will write the message to its local console. It would have to explicitly create a send operation to send anything back to the original sender
→ c. when foo() is called, it can match more than one message in the message queue - Yes, the receive calls foo() recursively, so after a successful match it will immediately execute the receive again
d. It will execute the receive only after waiting for 3 seconds - No, it will abort the receive if nothing arrives in the queue for 3 seconds
→ e. receive will find a match if the message contains a single tuple parameter and the first element of the tuple is an atom called wordse

4. In Erlang, what does the following expression represent?
   {27, 66}

a. record
b. record
c. set
→ d. tuple
e. Map

5. What is the name of the Erlang virtual machine?

a. erlc
→ b. beam
c. erl
d. cord
e. repl

6. When running an Erlang application, we can specify (on the command line) the function that should be run first. If we don't specify the initial function, erlang will default to...

a. It will simply run the code from top to bottom (of the initial module), so the first function will be whatever function is listed first in the source code.
b. erlang()
c. main()
→ d. start()
e. first()

7. Which of the following statements about Erlang tasks/threads are true? You may select more than one option)

→ a. Each Erlang task has its own message queue
b. Erlang threads are scheduled by the operating system (e., Linux or Windows) - No, they are scheduled by the Erlang VM
c. Java threads work the same way as Erlang tasks - No, definitely not. Java threads are quite "heavy" and typically are scheudled by the operating system. However, there is a third party Java framework called Akka that does use the Erlang model.
→ d. All Erlang threads must shared the same time slice that the operating system has given to the Erlang VM itself - Yes, this is true. The operating system allows the VM to run for a certain length of time, and the Erlang scheduler uses this for all tasks.

e. Erlang tasks are best suited to thread counts < 100, since a larger number would overwhelm the system - No, definitely not. Task counts > 100K are fine

8. When creating erlang records, names and keys must be...

a. ints
b. strings
c. tuples
→ d. atoms - Yes, these are basically string constants
e. Any valid erlang data type can be used for names and keys

9. Which of the following statements is true?

→ a. Only C explicitly defines the type - Yes, only C (of the languages in this course)
b. Both Erlang and Clojure explicitly define the type associated with a variable or label
c. Both Erlang and C explicitly define the type associated with a variable or label
d. Both Clojure and C explicitly define the type associated with a variable or label
e. Both Python and Erlang explicitly define the type associated with a variable or label

10. Erlang can support method overloading (i.e., multiple versions of a function that share the same name).

Like Java, Clojure, etc, this mechanism only works if each version of the method has a different number of parameters. Otherwise, it would be impossible to tell them apart.
Select one:
True
→ False
While it's true that Erlang can distinguish between versions of the same method, based upon the number of parameters, it can also support versions of the same method even if they all take the same number of parameters. It does this using pattern matching.

For example, the code below show three versions of foo that all take one tuple as the only parameter. But each version of the method is associated with a different atom in the first position of the tuple (dog, cat, or pig). So Erlang can match this to the tuple passed to the function to determine which version of the function to call. Other languages that we looked at in this course can not do that.

11. What, if anything, is wrong with the Erlang code below?

```
start() ->
    X = 2 + 4,
    Y = 6 * 2,
    Z = X + Y,
    io:fwrite("Z: ~w~n", [Z] ).
```

→ a. The code is fine, and the function will output 18 if it is run - Correct. This is a very basic Erlang function.
b. The first line should terminate with a :, not ->
c. The parameters should be listed in [ ], not ( ).

d. Each function expression should end with a ;
e. Each function expression should be surrounded by ( )

12. When exporting erlang modules, you must indicate the number of parameters associated with the function that is being exported.

Select one:
→ True
False
Yes, this form is unique to Erlang. An example is listed below. Here we are exporting three functions from the current module: a function named baz that takes one argument, and two versions of bar -- one that takes a single argument and one that takes two arguments.

```
-export([baz/1, bar/1, bar/2] ).
```

13. Which of the following statements is false? You can choose more than one option

→ a. listen() is an Erlang library function that causes the task to wait for a new message to arrive - False, there is no pre-defined listen function. The primary Erlang function is to receive
→ b. If no messages arrive after receive is invoked, we have a permanent deadlock - No. It will indeed wait for a new message, but there are timeout mechanisms that can be used to prevent permanent deadlock.
→ c. If Erlang finds a match with the first message in the queue, it will then move on to the second message to see if can match it as well. - No, by default receive will stop after finding a match. If we want to match many messages, we must call receive many times, or perhaps call the queue processing function recursively.
d. With erlang, we can define multiple receive patterns that can be matched against difference message types - This is true. It's what makes receive so powerful
e. When matching messages Erlang will start with the first message in the queue

14. What is the final line in following erlang code actually doing?

```
foo()->
  Thing = spawn(bar, baz,[4, 3]),
  Thing ! {self(), “Stuff”}.
```

a. Creating a tuple called Thing and adding it to the process registry
→ b. Sending a message to a new process identified as Thing, and including its own process identifier in the message - Correct. This is a common communication pattern. It allows the receiver to send a return message back to the parent thread.
c. Creating a process named Thing and asking it to send a message called Stuff back to the parent
d. This syntax doesn't make any sense for Erlang, so it would produce an error
e. Creating a new process and initializing it with a name "self" and a string called "stuff"

15. For basic math functions (e.h., addition, subtraction), which notation does Erlang use to express the logic of the arithmetic operation?

a. postfix
→ b. infix - Yes, 2 + 2

c. prefix
d. None of the other answers is correct
e. erfix

16. In Erlang, what is the return value of spawn?

a. either true or false to indicate success or failure
b. A list containing each of the new processes that have been created
c. The string name of task that was created - No, it's the numeric identifier of the new process
→ d. The numeric identifier of the new process
e. nil

17. Communication patterns in Erlang must represent a parent/child relationship. Otherwise, there would be no way for different processes to find one another.

Select one:
True
→ False
It's certainly true that parent/child process relationships can be used. However, this would be very restrictive in applications with tens of thousand of threads.
Instead, Erlang supports a registry mechanism that allows processes to add their name/id to a shared database that is accessible to all processes. So an arbitrary process only has to query the registry to get the contact information for any named process in the entire system. So no parent/child relationship is required.

18. Which composite data structure is found in Erlang but not in either Python or Clojure?

a. map/dictionary
b. vector
→ c. record - Correct, neither Python or Clojure has a record structure like Erlang
d. tuple
e. set