EMultiple Choice: 2 pts each CS-3160 Exam #3 (CH11-CH13) FORM: K

Choose the BEST answer of those given and enter your choice on the Answer Sheet. You may choose multiple options, but the point value will be divided by the number of options chosen.

#1 What is an 'abstraction'?
(a) A representation that includes only the most significant parts.
(b) A function or procedure that can work on multiple data types.
(c) A complex data structure build from simpler data structures.
(d) A generic representation that glosses over the details.
(e) A programmer defined data type that can be re-used.

#2 A functions or procedure can generally be classified as a
(a) code abstraction.
(b) process abstraction.
(c) access abstraction.
(d) data abstraction.
(e) referential abstraction.

#3 An abstract data type includes
(a) just the subprograms that act on a particular type of data.
(b) a representation of the data as well as the processes that act upon that data.
(c) any and all subprograms that ever interact with the data that defines that type.
(d) the data elements only -- subprograms are a different type of abstraction.
(e) the representation of a complex data concept, such as through the use of a 'struct' in C/C++.

#4 A primary motivator for data abstraction is to
(a) reduce code size by only storing relevant information.
(b) improve software efficiency and performance.
(c) manage complexity.
(d) provide accountability for program managers.
(e) leverage the capabilities of the CPU.

#5 A 'constructor' is a method that
(a) registers the new object with the memory manager's garbage collector.
(b) allocates memory for any heap-dynamic data members of the object.
(c) is called whenever a new instance of a class object is created.
(d) is called whenever an instance of a class object is deleted.
(e) initializes all data members to 'safe' values.

#6 Most floating-point data types qualify as an abstract data type for all of the following reasons except:

(a) the operations that can be performed on the data is largely independent of the internal representation of the data.

(b) programs that use floating-point data types can generally be compiled on any platform that supports floating point data types regardless of the internal representation.

(c) the user is not allowed to create new operations on the data unless they are built from existing operations.

(d) the internal representation of the data is hidden and largely inconsequential to most programmers.

(e) the data type has been standardized as part of the IEEE-754 floating point standard.

#7 To qualify as an abstract data type, all of the following must be true except:

(a) the type's interface does not depend on the internal representation of the objects or the implementation details of the operations.

(b) the internal representation must be clearly defined and documented.

(c) the only direct operations on the type are those that are part of the type's definition.

(d) the declaration of the the type and the protocols for the operations on that type are contained in a single syntactic unit.

(e) the internal representation of the data must be hidden from the program units that use the type.

#8 The calls to an object's subprograms are sometimes referred to as _____ to that object.

(a) invocations

(b) requests

(c) beacons

(d) recitations

(e) messages

#9 A 'destructor' is a method that

(a) is called whenever an instance of a class object is deleted.

(b) allocates memory for any heap-dynamic data members of the object.

(c) is called whenever a new instance of a class object is created.

(d) initializes all data members to 'safe' values.

(e) registers the new object with the memory manager's garbage collector.

#10 Which of the following is NOT an example of 'hidden concurrency'?

(a) Instruction and data prefetch queues.

(b) Instruction and data pipelining.

(c) Separate busses for instructions and data.

(d) Multicore processors.

(e) Arithmetic parallelism in execution.

#11 When access to internal data members is needed, accessor methods (getters and setters) are used for all of the following reasons except.

(a) the level of access can be controlled by limiting the availability of the accessor methods.

(b) accessor methods provide fast, efficient access to data members without the function call overhead that would otherwise be involved.

(c) data validation can be performed on the data before it is accepted.

(d) if the internal representation is changed, then only the accessor methods need be modified.

(e) the internal data representation can be altered while keeping the same interface to the outside world.

#12 The purpose of having parameterized abstract data types is

(a) to allow a generically defined data type be able to handle actual data of various types.

(b) to be able to create arrays of the abstract data type.

(c) to be able to pass initialization data to a constructor.

(d) to improve code readability by hiding implementation details.

(e) to allow the compiler to choose configuration settings.

#13 Providing a modified version of an inherited subprogram results in a(n) ______ subprogram.

(a) overridden

(b) superceded

(c) eclipsed

(d) hidden

(e) encapsulated

#14 In object-oriented terminology, a 'method' is a subprogram that

(a) is specific to a class and can only be invoked via an instance of an object of that class.

(b) acts only on one instance of a class.

(c) provides access to the internal data of a class instance.

(d) is specific to a class and can be invoked without requiring an instance of an object of that class.

(e) does not return a value.

#15 The entire collection of a subprograms associated with a class is known as the _____ of the object.

(a) back end

(b) capability list

(c) front end

(d) interface

(e) firewall

#16 Destructors are rarely used in C# because

(a) the garbage collector has the ability to release all memory resources used by an object.

(b) file and other resources are released by the garbage collector.

(c) destructors have proven to be security risks.

(d) the use of garbage collection makes the timing of when a destructor is executed unpredictable.

(e) all objects are heap-dynamic and therefore do not need to be destroyed.

#17 The common name used to describe paramaterized abstract data types in most languages that support them is

(a) 'abstracts'.

(b) 'generics'.

(c) 'flexibles'.

(d) 'concretes'.

(e) 'encapsulations'.

#18 The problem of different programmers calling their methods or classes the same thing is mitigated through the use of

(a) namespaces.

(b) libraries.

(c) name registries.

(d) name reference blocks.

(e) symbol tables.

#19 Code reuse that allows a programmer to use those portions of the original code while modifying other portions to fit their specific needs describes

(a) inheritance.

(b) objectivism.

(c) automorphism.

(d) code duality.

(e) polymorphism.

#20 What distinguishes an instance variable from a class variable?

(a) An object must be referenced when accessing a class variable.

(b) Each object has its own version of an instance variable while all objects of a class share a single version of a class variable.

(c) Instance variable have wider scope than class variables.

(d) Instance variables are private while class variables are public.

(e) Each object has its own version of an class variable while all instances of a class share a single instance of an instance variable.

#21 One problem us using inheritance to promote code reuse is that it
(a) hides information making it increasingly difficult to base new classes on existing ones.
(b) requires class designers to know the inner workings of the higher level classes.
(c) allows programmers to use code written by someone else.
(d) results in deep hierachies from multiple sources that must be coordinated.
(e) creates dependencies among classes that can complicate code maintainability.

#22 Dynamic binding of subprogram calls to subprogram definitions is needed in order to support
(a) platform independence.
(b) encapsulation.
(c) polymorphism.
(d) inheritance.
(e) envelopment.

#23 A class that defines subprogram protocols but not bodies for some or all of its subprograms is called a(n)
(a) referential class.
(b) abstract class.
(c) template.
(d) incomplete class.
(e) concrete class.

#24 A common way to mitigate the disadvantages associated with multiple inheritance is to use
(a) diamond inheritance.
(b) graphic inheritance.
(c) overridden subprogram.
(d) interfaces.
(e) parametric polymorphism.

#25 If objects are allowed to be stack-dynamic, what problem can arise as a result of subclass types?
(a) object slicing.
(b) circular references.
(c) over-defined objects.
(d) polymorphic inconsistency.
(e) under-defined objects.

#26 The motivation for allowing static binding of class subprogram calls is to improve
(a) writability.
(b) maintainability.
(c) security.
(d) reliability.
(e) performance.

#27 Which levels of concurrency pose challenges and issues for programming language design?
(a) instruction and program
(b) instruction and statement
(c) program and subprogram
(d) instruction and unit
(e) statement and subprogram

#28 The metric by which a concurrent algorithm is considered 'scalable' is if
(a) doubling the number of processors doubles the speed of execution.
(b) the computational complexity of the algorithm is linear, O(n), or better.
(c) the algorithm breaks the task into many different subtasks.
(d) the algorithm is recursive, since each level of recursion can be run on a separate processor.
(e) the speed of execution increases in rough proportion to the number of processors avaialble.

#29 What is required in order to have 'logical concurrency'?
(a) The program units that are to be run concurrently must be independent.
(b) The execution of the program units alternates back and forth so that both appear to be running at the same time.
(c) Multiple instructions are executing at the exact same time.
(d) The program task must be broken up and spread across multiple machines.
(e) More than one processor must be dedicated to a given program unit.

#30 The sequence of statements that a program executes is known as
(a) an execution path.
(b) an execution trajectory.
(c) a thread of control.
(d) a program unit.
(e) a subprogram.

#31 The basic four reasons for using concurrency include all of the following EXCEPT
(a) simpler program conceptualization for many complex programming tasks.
(b) the development of distribute programs.
(c) speed of execution on non-compute bound programs even on single processor machines.
(d) improving overall program efficiency on compute-bound single processor systems.
(e) speed of execution on multiprocessor systems.

#32 Which of the following distinguishes 'tasks' from typical subprograms?
(a) The caller of a task my be able to continue execution before the task completes.
(b) Upon completion of the task, control may not return to the caller.
(c) A task my be invoked implicitly.
(d) All of these.
(e) None of these.

#33 What distinguishes a heavyweight task from a lightweight task?
(a) Heavyweight tasks are active throughout program execution while lightweight tasks are only active when needed.
(b) Heavyweight tasks account for more than half of processor execution time whle lightweight tasks do not.
(c) Heavyweight tasks require memory access while lightweight tasks only use processor registers.
(d) Heavyweight tasks run in their own address space while lightweight tasks do not.
(e) Heavyweight tasks are callable from anywhere in the program while lightweight tasks must be called from object instances.

#34 Possible communication scenarios between tasks include all of the following EXCEPT
(a) shared runtime-stack variables.
(b) shared non-local variables.
(c) no communication at all.
(d) message passing.
(e) parameters.

#35 A task that does not communicate with or affect the execution of any other task in the program is said to be
(a) asynchronous.
(b) disjoint.
(c) distinct.
(d) autonomous.
(e) singular.

#36 A monitor can provide what type(s) of synchronization?
(a) Cooperation but not competition.
(b) Competition but not cooperation.
(c) Both competition and cooperation.
(d) Neither competition nor cooperation.
(e) Either competition or cooperation, but not both.

#37 A(n) ________ exists when the relative order of execution of different concurrent tasks results in different results.
(a) block
(b) semaphore fault
(c) access conflict
(d) deadlock
(e) race condition

#38 Which of the following is NOT a design issue for programming languages in terms of support for concurrency?
(a) How and when tasks are created.
(b) How synchronization is provided.
(c) How and when tasks start and end.
(d) How tasks are assigned to available processors.
(e) How task scheduling is performed.

#39 A relatively simple means of providing synchronization between concurrent tasks is the use of
(a) schedulers.
(b) message passing.
(c) monitors.
(d) semaphores.
(e) callback methods.

#40 Encapsulating the shared data structures with the access operations on them is the focus of
(a) callback methods.
(b) monitors.
(c) semaphores.
(d) message passing.
(e) schedulers.

#41 The notion of a 'rendezvous' is associated with which synchronization method?
(a) semaphores.
(b) monitors.
(c) message passing.
(d) schedulers.
(e) callback methods.

#42 One of the chief goals of using abstract data types is to promote
(a) object-oriented programming.
(b) polymorphism.
(c) information hiding.
(d) operator overloading.
(e) inheritance.

#43 A common means of providing indirect access to data elements in an abstract data type is the use of
(a) 'friend' functions.
(b) setter and getter functions.
(c) hidden properties.
(d) pointer dereferencing.
(e) public data elements.

#44 The 'principle of least privilege' basically states that
(a) classes should only hide those elements and methods that might be changed in the future.
(b) programmers should only use those methods that are made public by the top-level classes they use.
(c) class methods should be made available to any client that might benefit from them.
(d) program elements should only be able to access data and methods that they require.
(e) class data should be made available directly to the user to promote program efficiency.

#45 Why do few languages provide built-in operations for abstract data types?
(a) Because most languages are derived directly or indirectly from C, which did not provide such operations.
(b) Doing so would be too confusing for many programmers.
(c) Language grammar rules make it too difficult.
(d) This is a false statement -- most language treat abstract data types like any other in this regard.
(e) Few operations have broad applicability to user-defined data types.

#46 In most object oriented languages, a call to the method Bar() on an object of type FooChild that is assigned to a reference variable of type Foo

(a) will throw an exception because an object of one type was assigned to a variable of a different type.

(b) will call the Bar() method defined by FooChild class, provided class Foo has a Bar() method, otherwise a compile time error would have resulted.

(c) will call the Bar() method defined by the Foo class.

(d) will call the Bar() method defined by FooChild, even if no such method is defined in class Foo.

(e) will call the Bar() method defined by the Foo class unless the variable is downcast to FooChild first.

#47 If one class is a subtype (as opposed to a subclass) of another class, this means that

(a) any public function provided by the subtype class is also provided by the parent class.

(b) a variable of a subtype can appear anywhere a variable of the parent type can appear without causing a type error.

(c) the parent type's private functions are a strict subset of the subtype's private functions.

(d) the class that is a subtype has the exact same public interface as the parent class.

(e) contains at least the same instance variables as the parent class.

#48 SIMD instruction sets are particularly beneficial for

(a) distributed computing applications.

(b) multithreaded applications.

(c) statistical modelling.

(d) multi-processor machines.

(e) vector processors.

#49 Programs that have coroutines but no concurrent subprograms are

(a) do not exist since coroutines are, by definition, concurrent subprograms.

(b) are implemented as concurrent subprograms even on single-processor machines.

(c) sometime called quasi-concurrent, but still only have a single thread of control.

(d) are still concurrent programs since each coroutine can run as a separate thread.

(e) are concurrent only if they are run on a machine with multiple processors.

#50 A general linguistic device that allows a block of code to run only when a specific condition is true is called

(a) a monitor.

(b) a try block.

(c) a guard.

(d) an assertion.

(e) a semaphore.