*1.*  *Which of the following statements are true?*

***1.***  ***In socket-level programming, a client operation to send data requires a server operation to read it. The implementation of client and server at the socket-level is tightly synchronized.***

***2.***  ***RMI enables you to program at a higher level of abstraction. It hides the details of socket server, socket, connection, and sending or receiving data. It even implements a multithreading server under the hood, whereas with socket-level programming you have to explicitly implement threads for handling multiple clients.***

***3.***  ***RMI applications are scalable and easy to maintain. You can change the RMI server or move it to another machine without modifying the client program except for resetting the URL to locate the server.***

***4.***  ***RMI clients can directly invoke the server method, whereas socket-level programming is limited to passing values.***

*2.*  *To start an RMI registry, use \_\_\_\_\_\_\_\_\_\_\_\_ from the command window.*

*1.*  *rmiregistry*

*2.*  *rmiregistry 7000*

***3.***  ***start rmiregistry 7000***

***4.***  ***start rmiregistry***

*3.*  *To locate a remote object with a name t at port 7000 on host panda.armstrong.edu, use*

*1.*  *Remote remoteObj = Name.lookup("//panda.armstrong.edu:7000/t");*

*2.*  *Remote remoteObj = Name.lookup("rmi://panda.armstrong.edu:7000/t");*

***3.***  ***Remote remoteObj = Naming.lookup("rmi://panda.armstrong.edu:7000/t");***

*4.*  *Remote remoteObj = Name.lookup("http://panda.armstrong.edu:7000/t");*

*4.*  *RMI is about \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.*

*1.*  *passing objects between a server and a client*

*2.*  *java.lang.Cloneable*

***3.***  ***accessing remote objects and invoking methods from remote objects.***

*4.*  *passing primitive data between a server and a client*

*5.*  *Each remote object has a unique name identified by an URL with the protocol rmi as follows:*

*1.*  *http://host/name*

*2.*  *http://host:port/name*

***3.***  ***//host:port/name***

*4.*  *rmi://host:port/name*

*6.*  *Which provides the naming services for the server to register the object and for the client to locate the object.*

*1.*  *Server implementation*

*2.*  ***RMI Registry***

*3.*  *Server stub*

*4.*  *Server object interface*

*7.*  *Which is an object that resides on the server host, communicates with the stub and the actual server object.*

*1.*  *Server object interface*

*2.*  *Server stub*

***3.***  ***RMI Registry***

*4.*  *Server Skeleton*

*8.*  *\_\_\_\_\_\_\_\_\_\_\_\_is a class that implements the remote object interface.*

*1.*  *RMI Registry*

*2.*  *Server object interface*

*3.*  *Server stub*

***4.***  ***Server implementation***

*9.*  *To register a remote object o with a name t at port 7000 on host panda.armstrong.edu, use*

*1.*  *Name.rebind("rmi://panda.armstrong.edu:7000/t", o);*

*2.*  *Name.bind("rmi://panda.armstrong.edu:7000/t", o);*

***3.***  ***Naming.rebind("rmi://panda.armstrong.edu:7000/t", o);***

***4.***  ***Naming.bind("rmi://panda.armstrong.edu:7000/t", o);***

*10.A remote object must be an instance of \_\_\_\_\_\_\_\_\_\_\_\_\_\_.*

*1.*  *java.rmi.RemoteObject*

*2.*  *java.lang.Cloneable*

*3.*  ***java.rmi.Remote***

*4.*  *java.io.Serializable*

*11.\_\_\_\_\_\_\_\_\_\_\_\_ is a subinterface of java.rmi.Remote that defines the methods for the server object.*

*1.*  ***Server object interface***

*2.*  *RMI Registry*

*3.*  *Server implementation*

*4.*  *Server stub*

*12.Assume that the file named policy contains the permission for registering a remote object with an RMI registry. To run the program (e.g.,RegisterWithRMIServer) that registers a remote object with an RMI registry, use the command \_\_\_\_\_\_\_\_\_ from the command window.*

*1.*  *java RegisterWithRMIServer java ?Djava.security.policy=policy*

*2.*  *java RegisterWithRMIServer*

***3.***  ***java ?Djava.security.policy=policy RegisterWithRMIServer***

*4.*  *java ?Dpolicy=policy RegisterWithRMIServer*

*13.Which of the following statements are true when passing arguments in a remoter method call.*

***1.***  ***Local object types, such as java.lang.String, are also passed by value, but this is completely different from passing an object parameter in a local call. Any object can be used as a parameter in a remote call as long as it is serializable. The stub serializes the object parameter and sends it in a stream across the network. The skeleton deserializes the stream into an object.***

***2.***  ***Remote object types are passed differently from local objects. When a client invokes a remote method with a parameter of a remote object type, the stub of the remote object is passed. The server receives the stub and manipulates the parameter through it.***

***3.***  ***When a client invokes a remote method with parameters, passing the parameters is handled by the stub and the skeleton.***

***4.***  ***Primitive data types, such as char, int, double, or boolean, are passed by value like a local call.***

*14.Which is a utility that registers remote objects and provides naming services for locating objects.*

*1.*  *Server implementation*

*2.*  ***RMI Registry***

*3.*  *Server Skeleton*

*4.*  *Server object interface*

*15.Which is an object that resides on the client host and serves as a surrogate for the remote server object.*

*1.*  ***Server stub***

*2.*  *Server Skeleton*

*3.*  *Server implementation*

*4.*  *Server object interface*

*16.To encode arguments and invoke a method on a remote object an RMI client uses local-*

*1.*  *prototype*

*2.*  *skeleton*

*3.*  *all of them*

***4.***  ***stub***