Q.1.

```
sub output

{
Line 1: print "<ul>"

Line 2: $conn = mysql_connect("mysql.foo.org:412","kum","overmoon");
Line 3: mysql_select_db( "kum", $conn ); #selects a database

Line 4: $q = " SELECT * FROM main WHERE id > " . $_GET["id"]. ";";
Line 5: $res = mysql_query( $q, $conn);
Line 6: while( $row = mysql_fetch_assoc($res) )
Line 7:             {
Line 8:                         print "<li>".$row['description']."</li>";
Line 9:             }

Line 10: print "</ul><br><ul>";

Line 12: $q = "SELECT * FROM main WHERE id < " . $_GET["id"]. ";";
Line 13: $res = mysql_query( $q, $conn);
Line 14:            while( $row = mysql_fetch_assoc($res) )
Line 15:                        {
Line 16:                                    print "<li>".$row['description']."</li>";
Line 17:                        }

Line 18:            print "</ul>";
}
```

Coding Errors:

1. Semicolon is missing at Line 1; **after print "<ul>".**

2. Variables scope not defined (declared without my, local, our keyword).

Coding Practice mistakes:

1. Improper indentation & too many unwanted loop code.

2. Code repetition: Code from line 4 to line 9 is repeating at line 12 to line 17,

   The mentioned code can be customize by creating a new function (Reusable code).

3. Hard coded Datasource name, user name, password etc.

My coding style for the given code:

```
#  my  $qry = "SELECT * FROM main WHERE id <". $obj_cgi->param('id');";
#  my  $obj_cgi : variable for CGI object.
#  my  $str_DSN = variable for Data Source name.
#  my  $str_pwd = variable for Password.
#  my  $str_user = variable for user name.
#  my  $str_dbh = variable for database handler.
#  my  $class_obj = It's a object  class which holds function 'display_output' .

# Passing values to function:
$class_obj->display_output ($obj_cgi, $str_DSN, $str_user, $str_pwd, $qry);

sub  display_output

        {
                my  ($obj_cgi, $str_DSN, $str_user, $str_pwd, $qry)=(@_);

                # Connection to DB
                my  $str_dbh = DBI->connect($str_DSN, $str_user, $str_pwd) or warn "$DBI::errstr\n";

                 # Preparing query
                my  $sth = $str_dbh->prepare(qq{$qry})or die $str_dbh->errstr;

                 # Executing query
                $sth->execute;

                print '<ul>';

                while (my @row=$sth->fetchrow_array)
                        {
                             print  $obj_cgi->Tr({-style=>'color: black; font-size : 15px;', -align=>'left'},
                                                       $obj_cgi->li($row['description']))
                        };

                print '</ul>';
        }
```

Q.2. Say you need to regularly send pricing parameters from a central server to 40 different servers around the globe. Suggest a solution to this problem.

Ans: As per my experience we can implement site synchronization method this can be implemented by small automation using perl & SOAP protocol.

Here, we need to develop two scripts, of which one will run at Central server & generate the '**pricing_parameters_packets**', these packets will be then shipped via SOAP protocol to the distributed servers. The second script will run at distributed server & will accept the received packet & will notify the central server.

Q.3.: My Design methodology & Script description:

I have developed a Web-based application for 'Miniature_pricer' as per given requirement.
I usually follow modular programming strategy & even in the given example I have followed the same method.

I usually break my script in three basic modules i.e.

- Separate Class/Package for database related stuff.
- Separate Class/Package for developing user interface.
- Separate Class/Package for misc. operations, such as log generation, notification.

Let me describe my script here,
My script is consisting of:

- Code files: **miniature.cgi, db_class.pm**.
- Supporting xml: **Dbconnection.xml**.
- Database query file: **miniature.sql.**
- Page style (GUI): **style.css file & images**.

Now, let's start
miniature.cgi : This script provides a GUI for user for his input & displays him the final output on the page submission. This script uses pragma's (use strict & use warnings) & modules (CGI & db_class).

db_class.pm : This modules has few functions which are describe below

- **sub new () :** This function acts as constructor for this class**.**
- **sub databaseconnection() :** This function parses the Dbconnection.xml file & establishes the connection to database & provides Database handle**.**
- **sub http_server():** This function parses the Dbconnection.xml file & returns Http Server's Name/IP.
- **sub get_rates() :** This function fetches the rates for the requested days & returns.
- **Sub get_future_contract :** This function does the core part by doing calculation & returns the output.

Dbconnection.xml : This file has information about Database server & Web hosting server.
style.css : This is a called in miniature.cgi script for page style along with images.