1. First CHECKED_CPX_CALL - Creating x variables:

```
CHECKED_CPX_CALL(CPXnewcols, env, lp, 1, &obj, &lb, &ub, &xtype, &xname);
// Parameters:
// env    - CPLEX environment pointer
// lp     - CPLEX problem pointer
// 1      - Number of columns (variables) to create (1 at a time)
// &obj   - Pointer to objective coefficient (0.0 for x variables as they
don't appear in objective)
// &lb    - Pointer to lower bound (0.0 for x variables)
// &ub    - Pointer to upper bound (CPX_INFBOUND as x ∈ R+)
// &xtype - Pointer to variable type ('C' for continuous)
// &xname - Pointer to variable name ("x_i_j")
```

2. Second CHECKED_CPX_CALL - Creating y variables:

```
CHECKED_CPX_CALL(CPXnewcols, env, lp, 1, &obj, &lb, &ub, &ytype, &yname);
// Parameters:
// env    - CPLEX environment pointer
// lp     - CPLEX problem pointer
// 1      - Number of columns (variables) to create (1 at a time)
// &obj   - Pointer to objective coefficient (C[i][j] for y variables)
// &lb    - Pointer to lower bound (0.0 for y variables)
// &ub    - Pointer to upper bound (1.0 as y is binary)
// &ytype - Pointer to variable type ('B' for binary)
// &yname - Pointer to variable name ("y_i_j")
```

3. CHECKED_CPX_CALL for Constraint (10) - Flow Conservation:

```
CHECKED_CPX_CALL(CPXaddrows, env, lp, 0, 1, idx.size(), &rhs, &sense,
&matbeg, &idx[0], &coef[0], NULL, NULL);
// Parameters:
// env         - CPLEX environment pointer
// lp          - CPLEX problem pointer
// 0           - Number of new columns (0 as we're just adding constraints)
// 1           - Number of new rows (1 constraint at a time)
// idx.size()  - Number of nonzero coefficients in the constraint
// &rhs        - Pointer to right hand side value (1.0 for flow
conservation)
// &sense      - Pointer to constraint sense ('E' for equality)
// &matbeg     - Pointer to beginning position in the constraint matrix (0)
// &idx[0]     - Pointer to array of variable indices in this constraint
// &coef[0]    - Pointer to array of coefficients for those variables
```

```
// NULL         - No new column names
// NULL         - No new row names
```

4. CHECKED_CPX_CALL for Constraints (11)-(12) - Assignment Constraints:

```
CHECKED_CPX_CALL(CPXaddrows, env, lp, 0, 1, idx.size(), &rhs, &sense,
&matbeg, &idx[0], &coef[0], NULL, NULL);
// Parameters:
// env          - CPLEX environment pointer
// lp           - CPLEX problem pointer
// 0            - Number of new columns (0 as we're just adding constraints)
// 1            - Number of new rows (1 constraint at a time)
// idx.size()   - Number of nonzero coefficients in the constraint
// &rhs         - Pointer to right hand side value (1.0 for assignment
constraints)
// &sense       - Pointer to constraint sense ('E' for equality)
// &matbeg      - Pointer to beginning position in the constraint matrix (0)
// &idx[0]      - Pointer to array of y variable indices in this constraint
// &coef[0]     - Pointer to array of coefficients (all 1.0)
// NULL         - No new column names
// NULL         - No new row names
```

5. CHECKED_CPX_CALL for Constraint (13) - Linking Constraints:

```
CHECKED_CPX_CALL(CPXaddrows, env, lp, 0, 1, idx.size(), &rhs, &sense,
&matbeg, &idx[0], &coef[0], NULL, NULL);
// Parameters:
// env          - CPLEX environment pointer
// lp           - CPLEX problem pointer
// 0            - Number of new columns (0 as we're just adding constraints)
// 1            - Number of new rows (1 constraint at a time)
// idx.size()   - Number of nonzero coefficients (2 for each linking
constraint)
// &rhs         - Pointer to right hand side value (0.0 for linking
constraints)
// &sense       - Pointer to constraint sense ('L' for less than or equal)
// &matbeg      - Pointer to beginning position in the constraint matrix (0)
// &idx[0]      - Pointer to array of indices [x_ij, y_ij]
// &coef[0]     - Pointer to array of coefficients [1.0, -(N-1)]
// NULL         - No new column names
// NULL         - No new row names
```

6. CHECKED_CPX_CALL for Writing the Problem:

```
CHECKED_CPX_CALL(CPXwriteprob, env, lp, "tsp_optimized.lp", NULL);
// Parameters:
// env                  - CPLEX environment pointer
```

```
// lp                    - CPLEX problem pointer
// "tsp_optimized.lp"    - File name to write to
// NULL                  - File type (NULL means determine from extension)
```

7. CHECKED_CPX_CALL for Solving:

```
CHECKED_CPX_CALL(CPXmipopt, env, lp);
// Parameters:
// env - CPLEX environment pointer
// lp  - CPLEX problem pointer
```

8. CHECKED_CPX_CALL for Getting Objective Value:

```
CHECKED_CPX_CALL(CPXgetobjval, env, lp, &objval);
// Parameters:
// env     - CPLEX environment pointer
// lp      - CPLEX problem pointer
// &objval - Pointer to store objective value
```

9. CHECKED_CPX_CALL for Getting Variable Values:

```
CHECKED_CPX_CALL(CPXgetx, env, lp, &varVals[0], 0, n-1);
// Parameters:
// env         - CPLEX environment pointer
// lp          - CPLEX problem pointer
// &varVals[0] - Pointer to array to store variable values
// 0           - Index of first variable to get
// n-1         - Index of last variable to get
```

10. CHECKED_CPX_CALL for Writing Solution:

```
CHECKED_CPX_CALL(CPXsolwrite, env, lp, "drilling-tsp-optimized.sol");
// Parameters:
// env                          - CPLEX environment pointer
// lp                           - CPLEX problem pointer
// "drilling-tsp-optimized.sol" - File name to write solution to
```