

CASE STUDY

twenty three

NBA Lineup

case study
OVERVIEW

CS23.1	Application Overview and Model Development
CS23.2	Worksheets
CS23.3	User Interface
CS23.4	Procedures
CS23.5	Re-solve Options
CS23.6	Summary
CS23.7	Extensions

CS23.1 *Application Overview and Model Development*

This application allows the user to view statistics and make informed coaching decisions for a basketball team. It is specifically designed to aid a coach or sports fan of the Indiana Pacers basketball team. These statistics are calculated from recorded data for several different player lineups; they include games when the Pacers played several different teams in both the regular and playoff seasons.

This application performs the following analyses:

- The performance of various lineup scenarios in which certain players are in and certain players are out
- Individual player statistics about the total minutes played and the average rating against a selected team
- Game statistics for all combinations of a player with his teammates.

There is also an optimization option in this application that allows the user to determine the best lineup of players to play a selected team in any season.

CS23.1.1 **Model Definition and Assumptions**

We assume that the game data for the Indiana Pacers team is provided. This data, which is imported by the developer, is stored in a hidden sheet. (See Figure CS23.1.) The user does have the option to view this sheet.

For each game played, the data contains the following statistics: opposing team, game number, season (regular or playoffs), and lineup. For each game's lineup, these additional statistics are recorded: minutes the lineup played, rating the lineup received, and the five players in the lineup. The application options analyze these statistics.

Game Data			End	Back						
Opposing Team	Game Number	Season	Minutes	Rating	Player 1	Player 2	Player 3	Player 4	Player 5	
NJ	1180	Regular	4.07	-92.16	Artest	Croshere	Hardaway	Mercer	O'Neal	
NJ	1180	Regular	2.13	-6.86	Artest	Hardaway	Harrington	Mercer	O'Neal	
NJ	1180	Regular	1.3	34.68	Artest	Hardaway	Harrington	O'Neal	RMiller	
NJ	1180	Regular	0.45	293.23	Artest	Harrington	JFoster	Mercer	O'Neal	
NJ	1180	Regular	9.04	-12.18	Artest	Harrington	Mercer	O'Neal	Tinsley	
NJ	1180	Regular	19.3	14.01	Artest	Harrington	O'Neal	RMiller	Tinsley	
NJ	1180	Regular	1.35	13.83	Bender	Croshere	Hardaway	JFoster	Mercer	
NJ	1180	Regular	1.63	95.82	Bender	Croshere	Hardaway	Mercer	O'Neal	
NJ	1180	Regular	1.82	-127.41	Bender	Hardaway	Harrington	JFoster	Mercer	
NJ	1180	Regular	5.67	11.69	Harrington	JFoster	Mercer	O'Neal	Tinsley	
NJ	1180	Regular	1.25	45.05	Harrington	JFoster	O'Neal	RMiller	Tinsley	
NY	1172	Regular	0.02	-5.14	Artest	Croshere	Hardaway	JFoster	Mercer	
NY	1172	Regular	4.19	77.2	Artest	Croshere	Hardaway	JFoster	RMiller	
NY	1172	Regular	1.25	76.79	Artest	Croshere	Hardaway	Mercer	O'Neal	
NY	1172	Regular	2.37	57.3	Artest	Croshere	JFoster	Mercer	Tinsley	
NY	1172	Regular	6.33	13.53	Artest	Croshere	Mercer	O'Neal	Tinsley	
NY	1172	Regular	0.05	-2718.74	Artest	Hardaway	Harrington	JFoster	Mercer	
NY	1172	Regular	1.13	66.21	Artest	Hardaway	Harrington	Mercer	O'Neal	
NY	1172	Regular	2.77	-18.74	Artest	Hardaway	Harrington	O'Neal	RMiller	
NY	1172	Regular	12.3	-17.76	Artest	Harrington	O'Neal	RMiller	Tinsley	
NY	1172	Regular	1.95	-72.1	Artest	JFoster	O'Neal	RMiller	Tinsley	
NY	1172	Regular	3.3	-76.96	Bender	Brewer	Croshere	JFoster	Mercer	

Figure CS23.1 The hidden game data sheet.

The first analysis option is to view the performance of various lineup scenarios in which certain players are in and certain players are out of the lineup. To do so, the user selects

the lineup he or she wants to analyze by determining which players should be in the lineup and/or which players should be out of the lineup. In this way, the user does not have to know a specific lineup that was actually used, but rather how the team performs when certain players are in or out of the lineup. The user also selects a team and a season for which he or she wants to see the results of this lineup. For example, a coach may want to know how the team played against New York in the regular season when the player Artest was in the lineup but player RMiller was not. For this analysis, we loop over the statistics in the game data sheet to keep track of the total minutes and total rating for every row of data in which the team and season match the user's choice and in which the players that should be in the lineup are listed and the players that should be out of the lineup are not listed. The total minutes and average rating of the selected lineup are then calculated and displayed to the user. We use the following code to scan the data for this analysis.

```

Do While DataStart.Offset(row, 0).Value <> ""
  If Team = "All" Or Team = DataStart.Offset(row, 0).Value Then
    If Season = "Both" Or Season = DataStart.Offset(row, 2).Value Then
      Condition = True
      For i = 1 To NumIn
        InPlayer = InSet(i)
        If PlayerPresent(InPlayer, row) = False Then
          Condition = False
        End If
      Next i

      For i = 1 To NumOut
        OutPlayer = OutSet(i)
        If PlayerPresent(OutPlayer, row) = True Then
          Condition = False
        End If
      Next i

      If Condition = True Then
        TotalMinutes = TotalMinutes + DataStart.Offset(row, 3).Value
        TotalRatingMinutes = TotalRatingMinutes + DataStart.Offset(row, 3).Value * DataStart.Offset(row, 4).Value
      End If
    End If
    row = row + 1
  Loop

```

The second analysis option is to view individual player statistics for the total minutes played and the average rating against any selected team(s). For example, a user may want to know the total minutes and average rating that Bender has when playing against Atlanta and Boston. To perform this analysis, the game data is scanned and the total minutes and total rating are recorded for each team on the user's list in which the selected player is on the lineup. Then, the total minutes and average rating per team are displayed. We use the following code to scan the data for this analysis.

```

Do While DataStart.Offset(row, 1).Value <> ""
  For t = 1 To NumTeams
    If TeamName(t) = DataStart.Offset(row, 0).Value Then

```

```

    If PlayerPresent(InPlayer, row) Then
        TeamMinutes(t) = TeamMinutes(t) + DataStart.Offset(row, 3).Value
        TeamRatingMinutes(t) = TeamRatingMinutes(t) + _
            DataStart.Offset(row, 3).Value * DataStart.Offset(row, 4).Value
    End If
End If
Next t
row = row + 1
Loop

```

The third analysis option is to view the game statistics for all combinations of a player with his teammates. For example, a user may be interested to learn how well BMiller plays with JFoster and how well he plays with O'Neal; therefore, he or she would view the total minutes and average rating that were recorded when BMiller played with each of his teammates. To perform this analysis, we scan the game data and record the total minutes and total rating from every row in which the selected player and a particular teammate played together. The total minutes and average rating for each combination of the selected player and his teammates are then displayed. The application also presents the total minutes and average rating for which the selected player played for all of the game data. We use the following code to scan the data for this analysis.

```

Do While DataStart.Offset(row, 0).Value <> ""
    If PlayerPresent(SelectedPlayer, row) = True Then
        TotalMinutes = TotalMinutes + DataStart.Offset(row, 3).Value
        TotalRatingMinutes = TotalRatingMinutes _
            + DataStart.Offset(row, 3).Value * DataStart.Offset(row, 4).Value

        For p = 1 To NumPlayers
            NextPlayer = Player(p)
            If NextPlayer <> SelectedPlayer And _
                PlayerPresent(NextPlayer, row) = True Then
                    PlayerMinutes(p) = PlayerMinutes(p) + _
                        DataStart.Offset(row, 3).Value
                    PlayerRatingMinutes(p) = PlayerRatingMinutes(p) + _
                        DataStart.Offset(row, 3).Value * DataStart.Offset(row, 4).Value
            End If
        Next p
    End If
    row = row + 1
Loop

```

The final analysis option is lineup optimization. To determine the optimal lineup, which is the one with the best overall average rating, the user selects a team and a season. The application then reports the lineup with the maximum average rating that played against the selected team in the selected season.

To perform this analysis, we need to view the game data for each unique lineup of the selected team and season. In order to view the data for unique lineups, we first sort the game data by the players in the lineup. We begin by ensuring that each row's list of players is sorted alphabetically. Next, we perform a row sort with the "Orientation" argument of the "Sort" method:

```

Do While DataStart.Offset(row, 0).Value <> ""
    Range(DataStart.Offset(row, 5), DataStart.Offset(row, 9)).Sort
key1:=DataStart.Offset(row, 5), Order1:=xlAscending, Header:=xlGuess,
    OrderCustom:=1, MatchCase:=False, Orientation:=xlLeftToRight, _
    DataOption1:=xlSortNormal
    row = row + 1
Loop
    
```

We then sort the entire list of game data by each column of the lineup players. For example, if there are two rows with lineups “Bender, BMiller, Croshere, Mercer, Strickland” and “Bender, BMiller, Croshere, Mercer, Tinsley,” then the lineup ending with “Strickland” is sorted above the one ending with “Tinsley.” Since there are five columns of lineup players, and since Excel’s “Sort” method has a limit of three keys to sort by, we perform two separate sorts:

```

'first sort by last three players
Range(DataStart.Offset(1, 0), DataStart.End(xlDown).End(xlToRight)).Sort
key1:=DataStart.Offset(row, 7), order1:=xlAscending, _
key2:=DataStart.Offset(row, 8), order2:=xlAscending, key3:=DataStart.Offset(row, 9),
order3:=xlAscending, _
Header:=xlGuess, OrderCustom:=1, MatchCase:=False,
Orientation:=xlSortColumns, DataOption1:=xlSortNormal
    
```

```

'then sort by team name and first two players
Range(DataStart.Offset(1, 0), DataStart.End(xlDown).End(xlToRight)).Sort
key1:=DataStart.Offset(row, 5), order1:=xlAscending, _
key2:=DataStart.Offset(row, 6), order2:=xlAscending, _
Header:=xlGuess, OrderCustom:=1, MatchCase:=False,
Orientation:=xlSortColumns, DataOption1:=xlSortNormal
    
```

We can now scan the sorted data and copy the rows whose team name and season match those selected by the user for the analysis. We copy these rows to a hidden sheet used for the optimal lineup analysis. (See Figure CS23.2.) Notice in this figure how the lineups are sorted.

Optimal Layout Data		All Lineups for this Team and Season						
Team	Season	Minutes	Rating					
ATL	Regular	4.37	-15.69	Bender	BMiller	Croshere	Mercer	Strickland
		2.47	-13.26	Bender	BMiller	Croshere	Mercer	Tinsley
		0.90	-258.31	Bender	BMiller	Mercer	O'Neal	Tinsley
		0.98	-186.90	Bender	BMiller	O'Neal	RMiller	Tinsley
		0.35	-9.57	Bender	Croshere	JFoster	Mercer	Strickland
		0.28	160.85	Bender	Croshere	Mercer	O'Neal	Strickland
		3.88	98.38	Bender	Croshere	Mercer	O'Neal	Tinsley
		0.37	-5.47	Bender	Croshere	Mercer	RMiller	Strickland
		3.05	-28.28	BMiller	Croshere	Harrington	Mercer	Strickland
		0.68	-2.74	BMiller	Harrington	JFoster	Mercer	Strickland
		0.35	8.35	BMiller	Harrington	JFoster	O'Neal	Tinsley
		1.43	-58.62	BMiller	Harrington	Mercer	O'Neal	Strickland
		0.23	414.26	BMiller	Harrington	Mercer	O'Neal	Tinsley
		4.17	-49.25	BMiller	Harrington	O'Neal	RMiller	Strickland
		10.90	17.10	BMiller	Harrington	O'Neal	RMiller	Tinsley
		2.02	-50.34	Croshere	Harrington	Mercer	O'Neal	Strickland
		0.45	1.61	Croshere	Harrington	O'Neal	RMiller	Strickland
		0.00	6.79	Croshere	Harrington	O'Neal	RMiller	Tinsley
		11.12	31.62	Harrington	JFoster	O'Neal	RMiller	Tinsley

Figure CS23.2 The optimal layout hidden data sheet.

We now need to ensure that the analysis of each lineup is performed on unique lineups. In other words, if the lineup “Bender, BMiller, Croshere, Mercer, Strickland” occurred in more than one game, then we want to record the cumulative minutes played and the average rating received by this lineup. To do so, we use a function procedure that checks each lineup one row at a time and ensures that consecutive rows do not have perfectly matching players in each column:

```
Function SameSet(row)
  For col = 2 To 6
    If OptDataStart.Offset(row, col).Value <>
      OptDataStart.Offset(row - 1, col).Value Then
      SameSet = False
      Exit For
    Else
      SameSet = True
    End If
  Next col
End Function
```

We have now recorded the total minutes and total ratings for each unique lineup. We convert the total ratings to average ratings and list the unique lineups on the final optimal lineup sheet. We then sort all of these rows (with unique lineups, total minutes, and average rating) by the average rating in descending order. The optimal lineup, which is the first one listed, has the maximum average rating.

CS23.1.2 **Input**

The input for this application is the following:

- Lineup scenario analysis: the players in and out of the lineup, the team, and the season
- Player statistics: the player and the team
- All combinations analysis: the player
- Optimal lineup analysis: the team and the season

CS23.1.3 **Output**

The output for this application is the following:

- Lineup scenario analysis: the total minutes and the average rating for the selected lineup
- Player statistics: the total minutes and the average rating for the selected player and the team
- All combinations analysis: the total minutes and the average rating for each combination of the selected player and his teammates
- Optimal lineup analysis: the lineup with the maximum average rating for the selected team and the season

CS23.2 Worksheets

This application requires five main sheets: the welcome sheet, the lineup scenario sheet, the player statistics sheet, the all combinations analysis sheet, and the optimal lineup sheet. Note that we also incorporate the two hidden sheets described in the above section. The welcome sheet contains the title, the description of the application, and the “Start” button. (See Figure CS23.3.) The “Start” button displays the user options form.

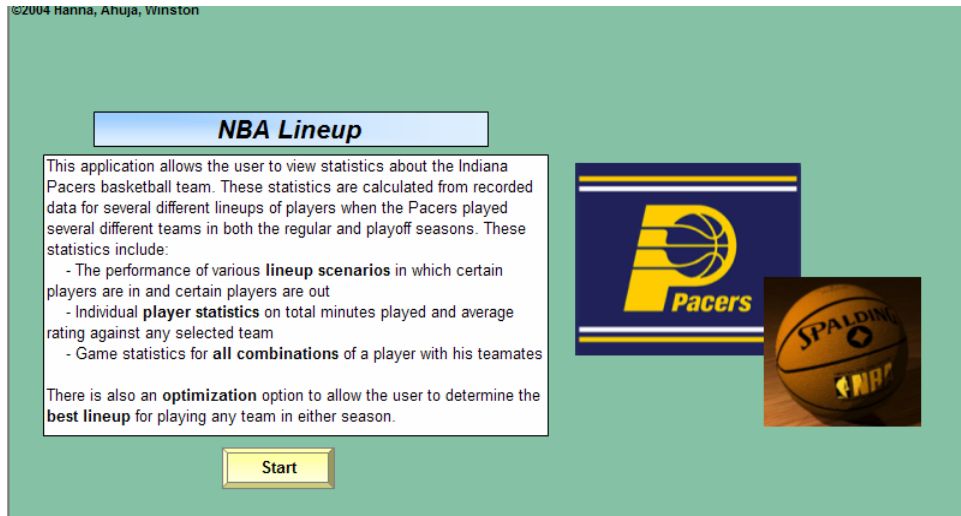


Figure CS23.3 The welcome sheet.

The lineup scenarios sheet presents the total minutes and the average rating that the selected lineup achieved when the Pacers played the selected team in the selected season. (See Figure CS23.4.) The selected lineup is determined by which players are in and which players are out of the lineup. The “End” button returns the user to the welcome sheet; the “Main Menu” button shows the user the options form again; the “New Lineup” button allows the user to re-solve this analysis option for a new set of input; and the “View Details” button allows the user to view the hidden game data sheet.

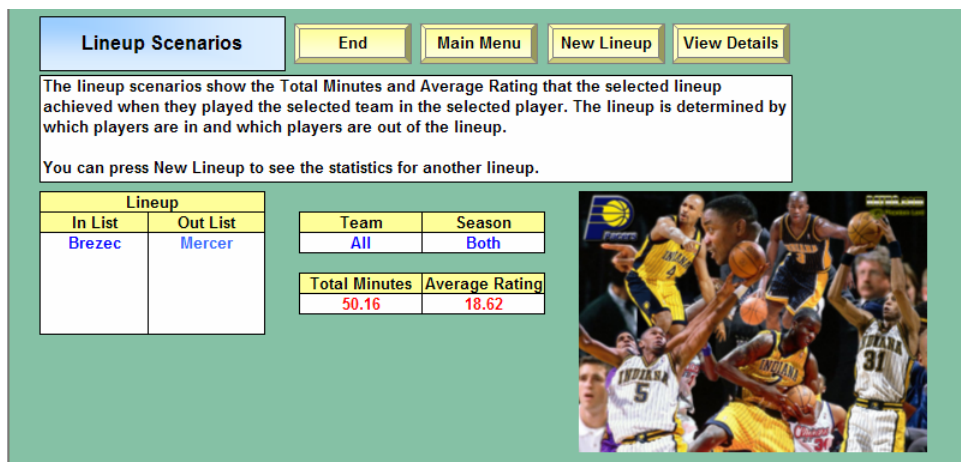


Figure CS23.4 The lineup scenarios sheet.

The player statistics sheet shows the total minutes played and average rating for the selected player when playing against each selected team(s). (See Figure CS23.5.) The “End,” “Main Menu,” and “View Details” buttons are also on this sheet. The “New Player” button allows the user to re-solve the player statistics option for a new player and/or the set of teams.

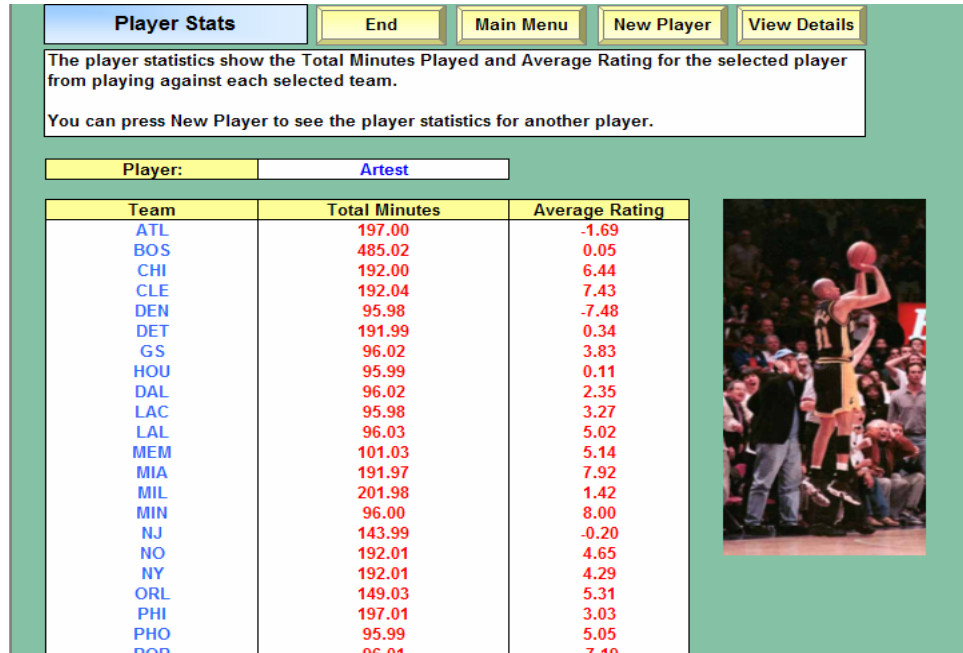


Figure CS23.5 The player statistics sheet.

The all combinations sheet displays the total minutes and total average rating of the selected player over all games as well as the total minutes and average rating for all games in which this player played with each of his teammates. (See Figure CS23.6.) The “End,” “Main Menu,” and “View Details” buttons are also on this sheet. The “New Player” button allows the user to re-solve the all the combinations analysis for a new player.

All Combinations
End
Main Menu
New Player
View Details

The all combinations option shows the Total Minutes and Total Average Rating of the selected player as well as the Total Minutes and Average Rating for all games in which they played with another player.

You can press New Player to see the statistics for all combinations with another player.

Player:	RMiller
---------	---------

Total Minutes	Total Average Rating
2297.91	5.39

Played With	Total Minutes	Average Rating
Artest	1454.64	8.49
Bender	152.96	-15.11
BMiller	1526.29	7.03
Brewer	30.09	30.09
Brezec	10.82	12.11
Croshere	177.08	-0.96
Hardaway	29.78	9.37
Harrington	1158.23	6.35
JFoster	226.65	-3.34
Mercer	565.64	5.31
O'Neal	1847.84	6.12
RMiller	-- --	-- --
Strickland	515.41	3.56
Tinsley	1481.98	3.70




Figure CS23.6 The all combinations sheet.

The optimal lineup sheet presents the optimal lineup for the selected team and season. (See Figure CS23.7.) The optimal lineup is the one with the best average rating. The application also reports the total minutes and average rating for each lineup that played against the selected team in the selected season. The “End,” “Main Menu,” and “View Details” buttons are also on this sheet. The “Re-solve” button allows the user to re-solve the optimal lineup analysis for a new team and/or season.

Optimal Lineup
End
Main Menu
Resolve
View Details

The optimal lineup is determined for the selected team and season. The optimal lineup is considered to be the lineup with the Best Rating. The Total Minutes and Average Rating for each lineup which played against the selected team in the selected season are also reported.

You can press Resolve to find the optimal lineup for another team and season.

Team	Season
ATL	Regular

Best Lineup					Total Minutes	Best Rating
BMiller	Harrington	Mercer	O'Neal	Tinsley	0.23	414.26

All Lineups for this Team and Season					Total Minutes	Average Rating
BMiller	Harrington	Mercer	O'Neal	Tinsley	0.23	414.26
Bender	Croshere	Mercer	O'Neal	Strickland	0.28	160.85
Harrington	JFoster	Jones	Mercer	Tinsley	0.62	146.06
BMiller	Mercer	O'Neal	RMiller	Tinsley	5.24	122.14
Bender	Croshere	Mercer	O'Neal	Tinsley	3.88	98.38
Artest	Brezec	Harrington	JFoster	Tinsley	1.69	58.81
Artest	BMiller	JFoster	Mercer	Strickland	1.72	55.42
Artest	BMiller	Harrington	JFoster	Strickland	3.75	44.12
Harrington	JFoster	O'Neal	RMiller	Tinsley	11.12	31.62
Artest	Croshere	Harrington	JFoster	Strickland	1.97	25.94
BMiller	Croshere	Harrington	Mercer	Tinsley	3.45	22.43
Artest	BMiller	Harrington	RMiller	Strickland	7.02	21.33
Artest	BMiller	Harrington	JFoster	Mercer	1.23	20.72
Artest	BMiller	Harrington	Mercer	O'Neal	9.40	18.78





Figure CS23.7 The optimal lineup sheet.

 Summary	Welcome sheet	Includes the application description and “Start” button.
	Lineup scenarios sheet	Reports the total minutes and average rating for the selected lineup.
	Player statistics sheet	Reports the total minutes and average rating for the selected player and team(s).
	All combinations sheet	Reports the total minutes and average rating for each combination of the selected player and his teammates.
	Optimal lineup sheet	Reports the lineup with the maximum average rating for the selected team and season.

CS23.3 *User Interface*

For this application’s user interface, we use navigational and functional buttons and six user forms. When the user presses the “Start” button on the welcome sheet, the options form appears. (See Figure CS23.8.) It allows the user to select which option to view: lineup scenarios, player statistics, all combinations, or optimal lineup. We use one frame with four option buttons for this form.

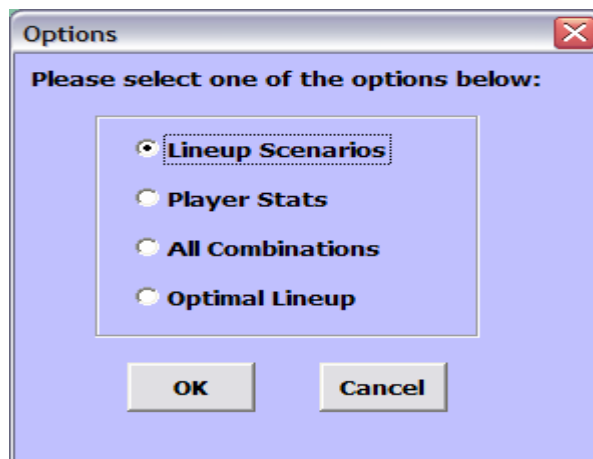


Figure CS23.8 The options form.

If the user chooses the lineup scenario option, the lineup scenario sheet appears and displays the lineup scenario form. (See Figure CS23.9.) On this form, the user selects which players should be in the lineup and which players should not be in the lineup for the analysis. The “In List” and “Out List” buttons allow the user to add the selected player from the list box on the left to the corresponding “In” or “Out” list box on the right. The “Remove” buttons remove a selected player from the boxes.

Lineup: Players

Please select a player and decide whether he should be in or out of the lineup. Select a player from the list and then click the In or Out button to place him in the list of selected players to be in or out of the lineup.

You can choose up to 5 players to be in (with 0 players selected to be out) and up to 5 players to be out (with 0 selected to be in). Any other combination of players selected to be in or out must total 5 or less.

Players:

- Artest
- Bender
- BMiller
- Brewer
- Brezec
- Croshere
- Hardaway
- Harrington
- JFoster
- Mercer
- O'Neal
- RMiller
- Strickland
- Tinsley

In List **Remove**

- Artest
- Brezec

Out List **Remove**

- Mercer

OK **Cancel**

Figure CS23.9 The lineup players form.

Once the user has specified the lineup, the lineup team and season form appears. (See Figure CS23.10.) We use a combo box to list all of the teams plus the “All” option to select all of the teams. We use another combo box to list the seasons plus the “Both” option to select both seasons.

If the user selects the player statistics option, then the player statistics sheet appears and displays the player statistics form. (See Figure CS23.11.) The user first selects the player from the combo box. Then, the user selects the teams for which he or she wants to see the player’s statistics. A list box provides all of the team names plus “All,” which the user can select to view the statistics for all the teams. The user can press the “Add” button to add a selected team to the team list box on the right. The “Remove” button removes a selected team from the list box.

Lineup: Team and Season

Teams
Please select the team or teams played for which you want to compare points scored.
Teams: All

Season
Please select which season you want to compare points scored.
Season: Both

OK Cancel

Figure CS23.10 The Lineup team and season form.

Players Stats

Please select for which player and teams you want to view summary stats.

Select a player from the list. Select a team from the list and click Add to add them to the list of Teams for which you want stats.

Player: Brewer

Teams:

All
ATL
BOS
CHI
CLE
DEN
DET
GS

Add Remove

ATL
CLE
DET

OK Cancel

Figure CS23.11 The player statistics form.

If the user selects the all combinations option, then the all combinations sheet appears and shows the all combinations form. (See Figure CS23.12.) On this form, the user selects a player from the combo box.

If the user selects the optimal lineup option, then the optimal lineup sheet appears and presents the optimal lineup form. (See Figure CS23.13.) On this form, the user selects the team and season for which he or she wants to find the optimal lineup. The user selects the team from the combo box and the season from another combo box.

All Combinations

Please select for which player you want to view rating combinations with other players.


Select a player from the list and click OK.

Players: Hardaway

OK Cancel

Figure CS23.12 The all combinations form.

Figure CS23.13 The optimal lineup form.

 <p>Summary</p>	Options form	The user chooses which analysis option to view.
	Lineup scenario form	The user selects which players should be in and out of the lineup to be analyzed.
	Lineup team and season form	The user selects the team and season for the lineup analysis option.
	Player statistics form	The user selects for which player and which teams to view the statistics.
	All combinations form	The user selects a player for the all combinations analysis.
	Optimal lineup form	The user selects the team and season for which to find the optimal lineup.
	Navigational buttons	“End,” “Main Menu,” “View Details”
	Functional buttons	“New Lineup,” “New Player,” “Re-solve”

CS23.4 *Procedures*

We will now outline the procedures for this application, beginning with the variable declarations and initial sub procedures. (See Figure CS23.14.) The “Main” procedure, which is called from the “Start” button, displays the options form to the user. The “Initialize” procedure sets several range variables that are used on the application’s worksheets.

```

Option Explicit
Option Base 1

Public i As Integer, p As Integer, t As Integer, row As Integer, col As Integer, _
DataStart As Range, StatsStart As Range, ScenStart As Range, AllCStart As Range, OptStart As Range, _
NumPlayers As Integer, NumTeams As Integer, Condition As Boolean, InPlayer As String, OutPlayer As String, _
Team As String, Season As String, SelectedPlayer As String, NextPlayer As String, NextTeamName As String, _
InSet() As String, OutSet() As String, NumIn As Integer, NumOut As Integer, Player() As String, TeamName() As String, _
TotalMinutes As Double, TotalRatingMinutes As Double, AverageRating As Double, _
PlayerMinutes() As Double, PlayerRatingMinutes() As Double, PlayerRating() As Double, _
TeamMinutes() As Double, TeamRatingMinutes() As Double, TeamRating() As Double, ws As Worksheet, Temp As Variant, _
NumLineups As Integer, LineupMinutes() As Double, LineupRatingMinutes() As Double, LineupRating() As Double, _
OptDataStart As Range

Sub MAIN() 'called from Start button
    frmOptions.Show
    'depending on option chosen, further input forms are shown
    'and corresponding procedure is called
End Sub

Sub Initialize()
    'initialize range variables for all sheets
    Set DataStart = Worksheets("Game Data").Range("B4")
    Set StatsStart = Worksheets("Stats").Range("B6")
    Set ScenStart = Worksheets("Scenarios").Range("B8")
    Set AllCStart = Worksheets("AllCombo").Range("B7")
    Set OptStart = Worksheets("Optimal").Range("B9")
    Set OptDataStart = Worksheets("Optimal Data").Range("B4")
End Sub

```

Figure CS23.14 The variable declarations and the “Main” and “Initialize” procedures.

The procedures for the options form are presented in Figure CS23.15. Depending on the selected option button, the corresponding sheet is made visible and the corresponding form appears.

```

Sub cmdCancel_Click()
    Unload Me
End Sub

Sub cmdOK_Click()
    Call Initialize

    If optScenario Then
        Worksheets("Scenarios").Visible = True
        Worksheets("Welcome").Visible = False
        Unload Me
        frmInput2.Show
    ElseIf optStats Then
        Worksheets("Stats").Visible = True
        Worksheets("Welcome").Visible = False
        Unload Me
        frmStats.Show
    ElseIf optAllCombo Then
        Worksheets("AllCombo").Visible = True
        Worksheets("Welcome").Visible = False
        Unload Me
        frmPlayer.Show
    ElseIf optOptimal Then
        Worksheets("Optimal").Visible = True
        Worksheets("Welcome").Visible = False
        Unload Me
        'frmOptimize.Show from Optimize procedure
        Call Optimize
    End If
End Sub

```

Figure CS23.15 The options form procedures.

The procedures for the lineup scenarios form are shown in Figure CS23.16. In Figure CS23.16 (a), the form is initialized by placing the list of players in the list box. The procedure for the “OK” button records the names of the players in the “In” list box and the “Out” list box to two respective arrays. The procedure in Figure CS23.16 (b) records the name of a selected player to the “In” list box. Several error checking statements are also included in this procedure. Similarly, in Figure CS23.16 (c), the procedure records the name of a selected player to the “Out” list box and also performs some error checking.

The procedures in Figure CS23.16 (d) remove a selected player from the “In” or “Out” list box, respectively.

```

Sub cmdCancel_Click()
Worksheets("Welcome").Visible = True
Worksheets("Scenarios").Visible = False
Unload Me
frmOptions.Show
End Sub

Private Sub UserForm_Initialize()
Range(ScenStart.Offset(1, 0), ScenStart.Offset(1, 1).End(xlDown)).ClearContents
ScenStart.Offset(1, 3).ClearContents
ScenStart.Offset(1, 4).ClearContents
Range(ScenStart.Offset(1, 6), ScenStart.Offset(1, 7)).ClearContents

NumIn = 0
NumOut = 0

lstAllPlayers.RowSource = "PlayersList"
lstAllPlayers.Value = lstAllPlayers.List(0)
lstAllPlayers.SetFocus
End Sub

Sub cmdOK_Click() 'from Input Form
If lstIn.ListCount = 0 And lstOut.ListCount = 0 Then
MsgBox "Please select at least one player for the lineup"
Exit Sub
End If

For p = 1 To NumIn
ScenStart.Offset(p, 0).Value = InSet(p)
Next p

For p = 1 To NumOut
ScenStart.Offset(p, 1).Value = OutSet(p)
Next p

Unload Me
frmInput3.Show
End Sub

```

Figure CS23.16 (a)

```

Sub cmdIn_Click()
'check array to ensure player has not already been added
For p = 1 To NumIn
If lstAllPlayers.Value = InSet(p) Then
MsgBox "You have already added this player. Please select another player."
Exit Sub
End If
Next p

'check other array to ensure player not added to other list
For p = 1 To NumOut
If lstAllPlayers.Value = OutSet(p) Then
MsgBox "You have already added this player to the Out List. Please select another player."
Exit Sub
End If
Next p

'increase array size and add team to array
NumIn = NumIn + 1
If (NumIn + NumOut) > 5 Then
MsgBox "You cannot have more than five players in the line up."
NumIn = NumIn - 1
Exit Sub
End If

ReDim Preserve InSet(NumIn)
InSet(NumIn) = lstAllPlayers.Value
lstIn.AddItem (lstAllPlayers.Value)
End Sub

```

Figure CS23.16 (b)

```

Sub cmdOut_Click()
    'check array to ensure team has not already been added
    For p = 1 To NumOut
        If lstAllPlayers.Value = OutSet(p) Then
            MsgBox "You have already added this player. Please select another player."
            Exit Sub
        End If
    Next p

    'check other array to ensure player not added to other list
    For p = 1 To NumIn
        If lstAllPlayers.Value = InSet(p) Then
            MsgBox "You have already added this player to the In List. Please select another player."
            Exit Sub
        End If
    Next p

    'increase array size and add team to array
    NumOut = NumOut + 1
    If (NumIn + NumOut) > 5 Then
        MsgBox "You cannot have more than five players in the line up."
        NumOut = NumOut - 1
        Exit Sub
    End If

    ReDim Preserve OutSet(NumOut)
    OutSet(NumOut) = lstAllPlayers.Value
    lstOut.AddItem (lstAllPlayers.Value)
End Sub

```

Figure CS23.16 (c)

```

Private Sub cmdRemoveIn_Click()
    If IsNull(lstIn.Value) Or lstIn.Value = "" Then
        MsgBox "Please select a player from the In List to remove."
        Exit Sub
    End If
    Temp = lstIn.Value
    For p = 1 To NumIn 'remove from array and re-adjust array values
        If InSet(p) = Temp Then
            If p = NumIn Then
                InSet(p) = ""
            Else
                InSet(p) = InSet(p + 1)
                For i = (p + 1) To (NumIn - 1)
                    InSet(i) = InSet(i + 1)
                Next i
            End If
        End If
    Next p
    NumIn = NumIn - 1
    ReDim Preserve InSet(NumIn)
    Exit For
End Sub

Private Sub cmdRemoveOut_Click()
    If IsNull(lstOut.Value) Or lstOut.Value = "" Then
        MsgBox "Please select a player from the Out List to remove."
        Exit Sub
    End If
    Temp = lstOut.Value
    For p = 1 To NumOut 'remove from array and re-adjust array values
        If OutSet(p) = Temp Then
            If p = NumOut Then
                OutSet(p) = ""
            Else
                OutSet(p) = OutSet(p + 1)
                For i = (p + 1) To (NumOut - 1)
                    OutSet(i) = OutSet(i + 1)
                Next i
            End If
        End If
    Next p
    NumOut = NumOut - 1
    ReDim Preserve OutSet(NumOut)
    Exit For
End Sub

Private Sub cmdRemoveOut_Click()
    If IsNull(lstOut.Value) Or lstOut.Value = "" Then
        MsgBox "Please select a player from the Out List to remove."
        Exit Sub
    End If
    Temp = lstOut.Value
    For p = 1 To NumOut 'remove from array and re-adjust array values
        If OutSet(p) = Temp Then
            If p = NumOut Then
                OutSet(p) = ""
            Else
                OutSet(p) = OutSet(p + 1)
                For i = (p + 1) To (NumOut - 1)
                    OutSet(i) = OutSet(i + 1)
                Next i
            End If
        End If
    Next p
    NumOut = NumOut - 1
    ReDim Preserve OutSet(NumOut)
    Exit For
End Sub

Private Sub cmdRemoveOut_Click()
    If IsNull(lstOut.Value) Or lstOut.Value = "" Then
        MsgBox "Please select a player from the Out List to remove."
        Exit Sub
    End If
    Temp = lstOut.Value
    For p = 1 To NumOut 'remove from array and re-adjust array values
        If OutSet(p) = Temp Then
            If p = NumOut Then
                OutSet(p) = ""
            Else
                OutSet(p) = OutSet(p + 1)
                For i = (p + 1) To (NumOut - 1)
                    OutSet(i) = OutSet(i + 1)
                Next i
            End If
        End If
    Next p
    NumOut = NumOut - 1
    ReDim Preserve OutSet(NumOut)
    Exit For
End Sub

```

Figure CS23.16 (d)

Figure CS23.16 The lineup players form procedures.

Figure CS23.17 presents the procedures for the lineup team and season form. In these procedures, the name of the selected team and season are recorded, and the “Lineup” procedure is then called.

```

Private Sub cmdCancel_Click()
    Worksheets("Welcome").Visible = True
    Worksheets("Scenarios").Visible = False
    Unload Me
    frmOptions.Show
End Sub

Private Sub cmdOK_Click()
    Team = cboTeams.Value
    Season = cboSeason.Value

    ScenStart.Offset(1, 3).Value = Team
    ScenStart.Offset(1, 4).Value = Season

    Unload Me
    Call Lineup
End Sub

Private Sub UserForm_Initialize()
    cboTeams.RowSource = "TeamsList"
    cboSeason.RowSource = "SeasonsList"
    cboTeams.Value = "All"
    cboSeason.Value = "Both"
End Sub

```

Figure CS23.17 The lineup team and season form procedures.

The “Lineup” procedure performs the analysis for the lineup scenario. (See Figure CS23.18.) The procedure loops over the statistics in the game data sheet to keep track of the total minutes and the total rating. It does so for every row of data in which the team and season match the user’s choice and in which the players that should be in the lineup are listed and the players that should be out of the lineup are not listed. The total minutes and average rating for the selected lineup are then calculated and displayed to the user.

We use a function procedure, “PlayerPresent,” to check if a player is in or out of the lineup. This function procedure appears in Figure CS23.19.

Figure CS23.20 presents the procedures for the player statistics form. In Figure CS23.20 (a), the procedures initialize the form and record the selected player; the “PlayerStats” procedure is then called. In Figure CS23.20 (b), the procedures record a selected team and add it to the team list box or remove a selected team from the team list box.

The “PlayerStats” procedure performs the analysis for the player statistics option. (See Figure CS23.21.) The game data is scanned, and the total minutes and total rating are recorded for each team on the user’s list in which the selected player is on the lineup. The total minutes and the average rating per team are displayed.

Figure CS23.22 presents the procedures for the all combinations form. In these procedures, the form is initialized and the selected player is recorded. The “AllComparisons” procedure is then called.

The “AllComparisons” procedure performs the analysis for the all combinations option. (See Figure CS23.23.) We scan the game data and record the total minutes and total rating from every row in which the selected player and a particular teammate have played together. The total minutes and average rating for each combination of the selected player and his teammates are displayed. The total minutes and average rating for which the selected player has played for all of the game data are also displayed.

```

Sub Lineup() 'Lineup Scenarios option
Worksheets("Scenarios").Activate
Application.ScreenUpdating = False

TotalMinutes = 0
TotalRatingMinutes = 0
row = 1
Do While DataStart.Offset(row, 0).Value <> "" 'scan game data to sum stats for this lineup
    If Team = "All" Or Team = DataStart.Offset(row, 0).Value Then
        If Season = "Both" Or Season = DataStart.Offset(row, 2).Value Then
            Condition = True
            For i = 1 To NumIn
                InPlayer = InSet(i)
                If PlayerPresent(InPlayer, row) = False Then
                    Condition = False
                End If
            Next i

            For i = 1 To NumOut
                OutPlayer = OutSet(i)
                If PlayerPresent(OutPlayer, row) = True Then
                    Condition = False
                End If
            Next i

            If Condition = True Then
                TotalMinutes = TotalMinutes + DataStart.Offset(row, 3).Value
                TotalRatingMinutes = TotalRatingMinutes + DataStart.Offset(row, 3).Value + DataStart.Offset(row, 4).Value
            End If
        End If
    End If
    row = row + 1
Loop

'output stats
If TotalMinutes = 0 Then
    MsgBox "This senario never occurred."
    Exit Sub
Else
    AverageRating = TotalRatingMinutes / TotalMinutes
    ScenStart.Offset(4, 3).Value = TotalMinutes
    ScenStart.Offset(4, 4).Value = AverageRating
End If
Application.ScreenUpdating = True
End Sub

```

Figure CS23.18 The “Lineup” procedure.

```

Function PlayerPresent(Player, row) 'function to search for player name in a row
    For col = 5 To 9
        If DataStart.Offset(row, col).Value = Player Then
            PlayerPresent = True
            Exit For
        End If
    Next col
End Function

```

Figure CS23.19 The “PlayerPresent” function procedure.

```

Sub cmdCancel_Click()
Worksheets("Welcome").Visible = True
Worksheets("Stats").Visible = False
Unload Me
frmOptions.Show
End Sub

Sub cmdOK_Click() 'from Input Form
If lstViewStats.ListCount = 0 Then
MsgBox "Please enter at least one team to view"
Exit Sub
End If

StatsStart.Offset(0, 1).Value = cboPlayerStats.Value
InPlayer = cboPlayerStats.Value

Unload Me
Call PlayerStats
End Sub

Sub UserForm_Initialize()
cboPlayerStats.RowSource = "PlayersList"
lstAllTeams.RowSource = "TeamsList"
cboPlayerStats.Value = cboPlayerStats.List(0)
lstAllTeams.Value = lstAllTeams.List(1)
lstAllTeams.SetFocus

NumTeams = 0
ReDim TeamName(NumTeams)
StatsStart.Offset(0, 1).ClearContents
Range(StatsStart.Offset(3, 0), StatsStart.Offset(3, 2).End(xlDown)).ClearContents
End Sub

```

Figure CS23.20 (a)

```

Private Sub cmdRemove_Click()
If IsNull(lstViewStats.Value) Or lstViewStats.Value = "" Then
MsgBox "Please select a team from your selected teams list to remove."
Exit Sub
End If
Temp = lstViewStats.Value
For t = 1 To NumTeams 'remove from array and re-adjust array values
If TeamName(t) = Temp Then
If t = NumTeams Then
TeamName(t) = ""
Else
TeamName(t) = TeamName(t + 1)
For i = t To NumTeams
TeamName(i) = TeamName(i + 1)
Next i
End If
NumTeams = NumTeams - 1
ReDim Preserve TeamName(NumTeams)
Exit For
End If
Next t
Temp = lstViewStats.ListIndex
lstViewStats.RemoveItem (Temp)
End Sub

Sub cmdView_Click()
'check array to ensure team has not already been added
For t = 1 To NumTeams
If lstAllTeams.Value = TeamName(t) Then
MsgBox "You have already added this team. Please select another team."
Exit Sub
End If
If TeamName(t) = "All" Then
MsgBox "You have already selected to view 'All' teams."
Exit Sub
End If
Next t

If lstAllTeams.Value = "All" And NumTeams > 0 Then
MsgBox "Please remove other names before selecting 'All'."
Exit Sub
End If

'increase array size and add team to array
NumTeams = NumTeams + 1
ReDim Preserve TeamName(NumTeams)

TeamName(NumTeams) = lstAllTeams.Value
lstViewStats.AddItem (lstAllTeams.Value)
End Sub

```

Figure CS23.20 (b)

Figure CS23.20 The player statistics form procedures.

```

Sub PlayerStats() 'Player Stats option
Worksheets("Stats").Activate
Application.ScreenUpdating = False

TotalMinutes = 0
TotalRatingMinutes = 0
If TeamName(1) = "All" Then
    NumTeams = 28
    ReDim TeamName(NumTeams)
    For t = 1 To NumTeams
        TeamName(t) = Range("TeamsList").Cells(t + 1, 1).Value
    Next t
End If

ReDim TeamMinutes(NumTeams), TeamRatingMinutes(NumTeams), TeamRating(NumTeams)
For t = 1 To NumTeams
    TeamMinutes(t) = 0
    TeamRatingMinutes(t) = 0
Next t

row = 1
Do While DataStart.Offset(row, 1).Value <> "" 'scan game data to sum stats for this player and team
    For t = 1 To NumTeams
        If TeamName(t) = DataStart.Offset(row, 0).Value Then
            If PlayerPresent(InPlayer, row) Then
                TeamMinutes(t) = TeamMinutes(t) + DataStart.Offset(row, 3).Value
                TeamRatingMinutes(t) = TeamRatingMinutes(t) + _
                    DataStart.Offset(row, 3).Value + DataStart.Offset(row, 4).Value
            End If
        End If
    Next t
    row = row + 1
Loop

'output stats
For t = 1 To NumTeams
    StatsStart.Offset(2 + t, 0).Value = TeamName(t)
    If TeamMinutes(t) = 0 Then
        StatsStart.Offset(2 + t, 1).Value = "Did not play against this team."
    Else
        TeamRating(t) = TeamRatingMinutes(t) / TeamMinutes(t)
        StatsStart.Offset(2 + t, 1).Value = TeamMinutes(t)
        StatsStart.Offset(2 + t, 2).Value = TeamRating(t)
    End If
Next t
Application.ScreenUpdating = True
End Sub

```

Figure CS23.21 The "PlayerStats" procedure.

```

Private Sub cmdCancel_Click()
    Worksheets("Welcome").Visible = True
    Worksheets("AllCombo").Visible = False
    Unload Me
    frmOptions.Show
End Sub

Private Sub cmdOK_Click()
    SelectedPlayer = cboPlayerAll.Value
    AlCStart.Offset(0, 1).Value = SelectedPlayer

    Unload Me
    Call AllComparisons
End Sub

Private Sub UserForm_Initialize()
    cboPlayerAll.RowSource = "PlayersList"
    cboPlayerAll.Value = cboPlayerAll.List(0)

    AlCStart.Offset(0, 1).ClearContents
    Range(AlCStart.Offset(3, 0), AlCStart.Offset(3, 1)).ClearContents
    Range(AlCStart.Offset(6, 1), AlCStart.Offset(19, 2)).ClearContents

End Sub

```

Figure CS23.22 The all combinations form procedures.

```

Sub AllComparisons() 'All Comparisons option
    Worksheets("AllCombo").Activate
    Application.ScreenUpdating = False

    TotalMinutes = 0
    TotalRatingMinutes = 0
    NumPlayers = 14
    ReDim Player(NumPlayers)
    For p = 1 To NumPlayers
        Player(p) = Range("PlayersList").Cells(p, 1).Value
    Next p
    ReDim PlayerMinutes(NumPlayers), PlayerRatingMinutes(NumPlayers), PlayerRating(NumPlayers)
    For p = 1 To NumPlayers
        PlayerMinutes(p) = 0
        PlayerRatingMinutes(p) = 0
    Next p

    row = 1
    Do While DataStart.Offset(row, 0).Value <> "" 'scan game data to sum stats for each player who played with the selected player
        If PlayerPresent(SelectedPlayer, row) = True Then
            TotalMinutes = TotalMinutes + DataStart.Offset(row, 3).Value
            TotalRatingMinutes = TotalRatingMinutes + DataStart.Offset(row, 3).Value + DataStart.Offset(row, 4).Value

            For p = 1 To NumPlayers
                NextPlayer = Player(p)
                If NextPlayer <> SelectedPlayer And PlayerPresent(NextPlayer, row) = True Then
                    PlayerMinutes(p) = PlayerMinutes(p) + DataStart.Offset(row, 3).Value
                    PlayerRatingMinutes(p) = PlayerRatingMinutes(p) + DataStart.Offset(row, 3).Value + DataStart.Offset(row, 4).Value
                End If
            Next p
        End If
        row = row + 1
    Loop

    'output stats
    AlCStart.Offset(3, 0).Value = TotalMinutes
    AverageRating = TotalRatingMinutes / TotalMinutes
    AlCStart.Offset(3, 1).Value = AverageRating
    For p = 1 To NumPlayers
        If Player(p) = SelectedPlayer Then
            AlCStart.Offset(p + 5, 1).Value = "---"
            AlCStart.Offset(p + 5, 2).Value = "---"
        Else
            If PlayerMinutes(p) = 0 Then
                AlCStart.Offset(p + 5, 1).Value = "Never played with this player"
            Else
                PlayerRating(p) = PlayerRatingMinutes(p) / PlayerMinutes(p)
                AlCStart.Offset(p + 5, 1).Value = PlayerMinutes(p)
                AlCStart.Offset(p + 5, 2).Value = PlayerRating(p)
            End If
        End If
    Next p
    Application.ScreenUpdating = True
End Sub

```

Figure CS23.23 The "AllComparisons" procedure.

The “Optimize” procedure performs the optimal lineup option. (See Figure CS23.24.) First, the optimal lineup form appears to record the team and season on which to perform the analysis. We then sort the game data by the players in the lineup. We begin by ensuring that each row’s list of players is sorted alphabetically. We then sort the entire list of game data by each column of the lineup players. We can now scan the sorted data and copy the rows whose team name and season match those selected by the user for the analysis. We copy these rows to the hidden optimal data sheet. Figure CS23.24 (b) illustrates these steps.

Next, we check each lineup one row at a time and ensure that consecutive rows do not have perfectly matching players in each column. We do so with the “SameSet” function procedure. (See Figure CS23.25.) We have now recorded the total minutes and total ratings for each unique lineup. We convert the total ratings to average ratings and list the unique lineups to the final optimal lineup sheet. We then sort all of these rows (with unique lineups, total minutes, and average rating) by the average rating in descending order. The optimal lineup is the first one listed; it has the maximum average rating. These steps are illustrated in Figure CS23.24 (b).

```

Sub Optimize() 'Optimal Layout option
Call Initialize
frmOptimize.Show
If frmOptimize.OptCont = False Then
Exit Sub
End If
Worksheets("Optimal").Activate
Application.ScreenUpdating = False

'sort names in each row
row = 1
Do While DataStart.Offset(row, 0).Value <> ""
Range(DataStart.Offset(row, 5), DataStart.Offset(row, 9)).Sort key1:=DataStart.Offset(row, 5), _
order1:=xlAscending, Header:=xlGuess, _
OrderCustom:=1, MatchCase:=False, Orientation:=xlLeftToRight, _
DataOption1:=xlSortNormal
row = row + 1
Loop

'sort all rows
'since only 3 keys allowed with Sort method, first sort by last three players
Range(DataStart.Offset(1, 0), DataStart.End(xlDown).End(xlToRight)).Sort key1:=DataStart.Offset(row, 7), order1:=xlAscending, _
key2:=DataStart.Offset(row, 8), order2:=xlAscending, key3:=DataStart.Offset(row, 9), order3:=xlAscending, _
Header:=xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlSortColumns, DataOption1:=xlSortNormal
'then sort by team and first two players
Range(DataStart.Offset(1, 0), DataStart.End(xlDown).End(xlToRight)).Sort key1:=DataStart.Offset(row, 5), order1:=xlAscending, _
key2:=DataStart.Offset(row, 6), order2:=xlAscending, _
Header:=xlGuess, OrderCustom:=1, MatchCase:=False, Orientation:=xlSortColumns, DataOption1:=xlSortNormal

'compare ratings of all lineups
ReDim LineupMinutes(1), LineupRatingMinutes(1), LineupRating(1)
LineupMinutes(1) = 0
LineupRatingMinutes(1) = 0
LineupRating(1) = 0

row = 1
i = 1
Do While DataStart.Offset(row, 0).Value <> "" 'copy data for selected team and season
If Team = DataStart.Offset(row, 0).Value Then
If Season = DataStart.Offset(row, 2).Value Then
Range(DataStart.Offset(row, 3), DataStart.Offset(row, 9)).Copy
OptDataStart.Offset(3 + i, 0).PasteSpecial xlPasteValues
i = i + 1
End If
End If
row = row + 1
Loop

```

Figure CS23.24 (a)

```

row = 4
NumLineups = 0
Do While OptDataStart.Offset(row, 0).Value <> "" 'collect stats
    If SameSet(row) = False Then
        'copy information and update array and insert new values
        NumLineups = NumLineups + 1
        Range(OptDataStart.Offset(row, 2), OptDataStart.Offset(row, 6)).Copy
        OptStart.Offset(NumLineups + 6, 0).PasteSpecial xlPasteValues

        ReDim Preserve LineupMinutes(NumLineups), LineupRatingMinutes(NumLineups)
        LineupMinutes(NumLineups) = LineupMinutes(NumLineups) + OptDataStart.Offset(row, 0).Value
        LineupRatingMinutes(NumLineups) = LineupRatingMinutes(NumLineups) + _
            OptDataStart.Offset(row, 0).Value * OptDataStart.Offset(row, 1).Value
    Else
        'update array value for previously recorded lineup (this is last entry in array due to sorting)
        LineupMinutes(NumLineups) = LineupMinutes(NumLineups) + OptDataStart.Offset(row, 0).Value
        LineupRatingMinutes(NumLineups) = LineupRatingMinutes(NumLineups) + _
            OptDataStart.Offset(row, 0).Value * OptDataStart.Offset(row, 1).Value
    End If
    row = row + 1
Loop

'output stats
If NumLineups = 0 Then
    MsgBox "No lineups were found for this team and season." & vbCrLf & _
        "(Note that there are only a few teams in the playoffs.)"
    Exit Sub
End If
ReDim LineupRating(NumLineups)
For p = 1 To NumLineups
    If LineupMinutes(p) = 0 Then
        LineupRating(p) = LineupRatingMinutes(p)
    Else
        LineupRating(p) = LineupRatingMinutes(p) / LineupMinutes(p)
    End If
    OptStart.Offset(p + 6, 5).Value = LineupMinutes(p)
    OptStart.Offset(p + 6, 6).Value = LineupRating(p)
Next p

'find best lineup (sort by rating and take first lineup = max rating value)
Range(OptStart.Offset(7, 6), OptStart.Offset(7, 6).End(xlDown)).Name = "AllLineups"
Range(OptStart.Offset(7, 0), OptStart.Offset(7, 6).End(xlDown)).Sort key1:=OptStart.Offset(7, 6), order1:=xlDescending
Range(OptStart.Offset(7, 0), OptStart.Offset(7, 6)).Copy
OptStart.Offset(4, 0).PasteSpecial xlPasteValues
OptStart.Select
Application.CutCopyMode = False
Application.ScreenUpdating = True
End Sub

```

Figure CS23.24 (b)
Figure CS23.24 The “Optimize” procedure.

```

Function SameSet(row) 'check if lineups are in same set
    'check lineup
    For col = 2 To 6
        If OptDataStart.Offset(row, col).Value <> OptDataStart.Offset(row - 1, col).Value Then
            SameSet = False
            Exit For
        Else
            SameSet = True
        End If
    Next col
End Function

```

Figure CS23.25 The “SameSet” function procedure.

Figure CS23.26 presents the procedures for the optimal lineup form. In these procedures, the form is initialized and the selected team and season are recorded.

```

Public OptCont As Boolean

Private Sub cmdCancel_Click()
    Worksheets("Welcome").Visible = True
    Worksheets("Optimal").Visible = False
    Unload Me

    OptCont = False
    frmOptions.Show
End Sub

Private Sub cmdOK_Click()
    If cboTeams.Value = "All" Then
        MsgBox "Please just choose one team. The 'All Teams' option will be available later."
        Exit Sub
    End If

    If cboSeason.Value = "Both" Then
        MsgBox "Please just choose one season. The 'Both Seasons' option will be available later."
        Exit Sub
    End If

    Team = cboTeams.Value
    Season = cboSeason.Value
    OptStart.Offset(1, 0).Value = Team
    OptStart.Offset(1, 1).Value = Season
    'optional on hidden data sheet:
    OptDataStart.Offset(1, 0).Value = Team
    OptDataStart.Offset(1, 1).Value = Season

    Unload Me 'return to Optimize procedure
End Sub

Private Sub UserForm_Initialize()
    cboTeams.RowSource = "TeamsList"
    cboSeason.RowSource = "SeasonsList"
    cboTeams.Value = cboTeams.List(1)
    cboSeason.Value = cboSeason.List(1)

    OptStart.Offset(1, 0).ClearContents
    OptStart.Offset(1, 1).ClearContents

    Range(OptStart.Offset(4, 0), OptStart.Offset(4, 9)).ClearContents
    Range(OptStart.Offset(7, 0), OptStart.Offset(7, 9).End(xlDown)).ClearContents
    Range(OptDataStart.Offset(4, 0), OptDataStart.Offset(4, 6).End(xlDown)).ClearContents
End Sub

```

Figure CS23.26 The optimize form procedures.

Figure CS23.27 presents the navigational procedures.

```

.....
'navigational procedures
.....

Sub MainMenu()
    Worksheets("Welcome").Visible = True
    ActiveSheet.Visible = False
    frmOptions.Show
End Sub

Sub ViewDetails()
    Set ws = ActiveSheet
    Worksheets("Game Data").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub EndProg()
    Worksheets("Welcome").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub AnotherLineup()
    Call Initialize
    frmInput2.Show
End Sub

Sub MoreStats()
    Call Initialize
    frmStats.Show
End Sub

Sub AnotherPlayer()
    Call Initialize
    frmPlayer.Show
End Sub

Sub GoBack()
    ws.Visible = True
    Worksheets("Game Data").Visible = False
End Sub


```

Figure CS23.27 The navigational procedures.

 <p>Summary</p>	Main	Initializes the application and shows the user the options form.
	Options form procedures	Determine which option the user wants to solve, takes him or her to the corresponding sheet, and displays a corresponding form.
	Lineup form procedures	Record which player should be in and out of the lineup.
	Lineup team and season form procedures	Record the team and season for the lineup analysis.
	Lineup	Scans the game data to report the total minutes and average rating that the specified lineup would have.
	PlayerPresent function	Determines if a player is in the lineup of a row in the game data sheet.
	Player statistics form procedures	Record the player and teams for the analysis.
	PlayerStats	Scans the game data to find the total minutes and average rating that the player had while playing the selected teams.
	All combinations form procedures	Record the player for the analysis.
	AllCombinations	Scans the game data to report the total minutes and average rating recorded when the selected player played with each of his teammates.
	Optimize	Determines the optimal lineup by sorting and scanning the game data of unique lineups for the selected team and season.
	Optimal lineup form procedure	Records the team and season for the analysis.
	SameSet function	Determines if two lineups are unique.
Navigational procedures	For the “End,” “Main Menu,” “View Details,” and “Re-solve” buttons.	

CS23.5 *Re-solve Options*

The user can re-solve this application with the various re-solve buttons on each of the four main sheets: the lineup sheet, the player statistics sheet, the all combinations sheet, and the optimal lineup sheet. These buttons are “New Lineup” on the lineup sheet, “New Player” on the player statistics and all combinations sheets, and “Re-solve” on the optimal lineup sheet. By selecting these buttons, the user can re-enter the parameters for each analysis performed.

 <p>Summary</p>	“New Lineup”	The re-solve button on the lineup sheet; the user can select a new lineup scenario, team, or season.
	“New Player”	The re-solve button on the player statistics and the all combinations sheet; the user can select a new player and a set of teams for the player statistics option; the user can also select a new player for the all combinations option.
	“Re-solve”	The re-solve button on the optimal lineup sheet; the user can select a new team or season for the analysis.

CS23.6 *Summary*

- This application allows the user to view statistics and make informed coaching decisions for a basketball team. The user can analyze lineup scenarios, player statistics, combinations of a player with his teammates, or find the optimal lineup to play a specific team in a specific season.
- This application requires five main sheets: the welcome sheet, the lineup scenario sheet, the player statistics sheet, the all combinations analysis sheet, and the optimal lineup sheet. Note that we also use two hidden sheets: the game data and the optimal lineup data.
- For this application's user interface, we use navigational and functional buttons and six user forms.
- Several of this application's procedures perform the analysis for the selected option.
- The user can re-solve this application by pressing the re-solve buttons on the analysis sheets to return to the corresponding input forms.

CS23.7 *Extensions*

- Add summary statistics to the player statistics option.
- Add summary statistics to the all combinations option.
- Add summary statistics to the optimal lineup option.
- Allow the user to import new game data into the application. Allow the user to either overwrite the current data or to append it.
- Create more analysis options for the optimal lineup option. What else would a coach be interested in examining when determining the optimal lineup? For example, what if a coach wants to know what the optimal lineup would be for a given team and season if he or she knows a particular player cannot be in the lineup?