```
Sub InputData(ByRef principal As Double,
              ByRef yearlyRate As Double, ByRef numMonths As Integer)
  'Input loan amount, yearly rate of interest, and duration
  Dim percentageRate As Double, numYears As Integer
  principal = CDbl(txtPrincipal.Text)
  percentageRate = CDbl(txtYearlyRate.Text)
  yearlyRate = percentageRate / 100   'Convert interest rate to decimal form
  numYears = CInt(txtNumYears.Text)
  numMonths = numYears * 12           'Duration of loan in months
End Sub

Function Payment(ByVal principal As Double, ByVal monthlyRate As Double,
                 ByVal numMonths As Double) As Double
  Dim estimate As Double          'Estimate of monthly payment
  estimate = principal * monthlyRate / (1 - (1 + monthlyRate) ^ (-numMonths))
  'Round the payment up if there are fractions of a cent
  If estimate = Math.Round(estimate, 2) Then
    Return estimate
  Else
    Return Math.Round(estimate + 0.005, 2)
  End If
End Function

Function GenerateMonthsArray(ByVal principal As Double,
                             ByVal monthlyRate As Double,
                             ByVal numMonths As Integer) As Month()
  Dim months(numMonths - 1) As Month
  'Fill the months array
  Dim monthlyPayment As Double = Payment(principal, monthlyRate, numMonths)
  'Assign values for first month
  months(0).number = 1
  months(0).interestPaid = monthlyRate * principal
  months(0).principalPaid = monthlyPayment - months(0).interestPaid
  months(0).endBalance = principal - months(0).principalPaid
  'Assign values for interior months
  For i As Integer = 1 To numMonths - 2
    months(i).number = i + 1
    months(i).interestPaid = monthlyRate * months(i - 1).endBalance
    months(i).principalPaid = monthlyPayment - months(i).interestPaid
    months(i).endBalance = months(i - 1).endBalance - months(i).principalPaid
  Next
  'Assign values for last month
  months(numMonths - 1).number = numMonths
  months(numMonths - 1).interestPaid = monthlyRate *
                                       months(numMonths - 2).endBalance
  months(numMonths - 1).principalPaid = months(numMonths - 2).endBalance
  months(numMonths - 1).endBalance = 0
  Return months
End Function
```

CHAPTER 7 SUMMARY

1. For programming purposes, lists of data are most efficiently processed if stored in an *array*. An array is declared with a *Dim* statement, which also can specify its *size* and initial values. The size of an already declared array can be specified or changed with a *ReDim* or *ReDim*

Preserve statement. The methods *Count*, *First*, *Last*, *Max*, and *Min* return the size of the array, first element, last element, largest element, and smallest element, respectively. For numeric arrays, the methods *Average* and *Sum* return the average and total of the numbers in the array.

2. The *IO.File.ReadAllLines* method returns a string array containing the contents of a file.
3. The *Split* method converts a line consisting of strings separated by a delimiter (usually a comma or a blank space) to a string array. The *Join* function is its inverse.
4. A *For Each loop* repeats a group of statements for each element in an array.
5. *LINQ* is a powerful Microsoft technology that provides a standardized way to set criteria for information retrieval from data sources, including arrays. Operators such as *From*, *Where*, *Distinct*, *Order By*, *Let*, and *Select* are used to create a *query expression* that can retrieve a list of information from the data source. When each element of the list is a single value, the *ToArray* method converts the list to an array. LINQ provides an easy way to sort the contents of an array.
6. The DataSource property can be used to display the sequence returned by a query. If the Select clause contains a single expression, a statement of the form

```
lstOutput.DataSource = query.ToList
```

displays the sequence in a list box. If the select clause contains two or more expressions, a statement of the form

```
dgvOutput.DataSource = query.ToList
```

displays the information returned by the query as a table in a DataGridView control.

7. The *binary search* method provides an efficient way to look for an element of an ordered array.
8. A *structure* is a composite programmer-designed data type with a fixed number of members, each of which can be of any data type. LINQ can be used to sort and search structures and to create new structures of *anonymous* data types.
9. A table can be effectively stored and accessed in a *two-dimensional array*.

CHAPTER 7 PROGRAMMING PROJECTS

1. Table 7.16 contains some lengths in terms of feet. Write a program that displays the nine different units of measure; requests the unit to convert from, the unit to convert to, and the quantity to be converted; and then displays the converted quantity. A typical outcome is shown in Fig. 7.37 on the next page.

TABLE 7.16 Equivalent lengths.

1 inch = .0833 foot	1 rod = 16.5 feet
1 yard = 3 feet	1 furlong = 660 feet
1 meter = 3.28155 feet	1 kilometer = 3281.5 feet
1 fathom = 6 feet	1 mile = 5280 feet

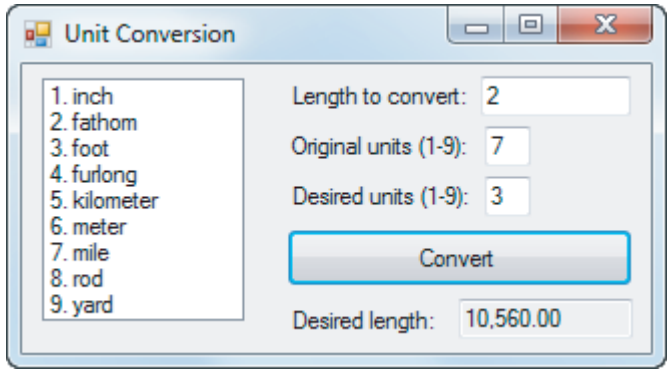

FIGURE 7.37 **Possible outcome of Programming Project 1.**

2. Statisticians use the concepts of **mean** and **standard deviation** to describe a collection of data. The mean is the average value of the items, and the standard deviation measures the spread or dispersal of the numbers about the mean. Formally, if $x_1, x_2, x_3, \ldots, x_n$, is a collection of data, then

$$\text{mean} = m = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

and standard deviation $=$

$$s = \sqrt{\frac{(x_1 - m)^2 + (x_2 - m)^2 + (x_3 - m)^2 + \cdots + (x_n - m)^2}{n}}$$

The file Scores.txt contains exam scores. The first four lines of the file hold the numbers 59, 60, 65, and 75. Write a program to calculate the mean and standard deviation of the exam scores, assign letter grades to each exam score, ES, as follows, and then display a list of the exam scores along with their corresponding grades, as shown in Fig. 7.38.

$ES \geq m + 1.5s$	A
$m + .5s \leq ES < m + 1.5s$	B
$m - .5s \leq ES < m + .5s$	C
$m - 1.5s \leq ES < m - .5s$	D
$ES < m - 1.5s$	F

For instance, if m were 70 and s were 12, then grades of 88 or above would receive A's, grades between 76 and 87 would receive B's, and so on. A process of this type is referred to as *curving grades*.

3. *Rudimentary Translator.* Table 7.17 gives English words and their French and German equivalents. These words have been placed into the file Dictionary.txt. The first two lines of the file are

```
YES,OUI,JA
TABLE,TABLE,TISCH
```

Write a program that requests an English sentence as input and translates it into French and German. Assume that the only punctuation in the English sentence is a period at the end of the sentence. (***Note:*** If a word in the sentence is not in the dictionary, it should appear as itself in the French and German translations. This will allow proper nouns to be translated correctly.) See Fig. 7.39.

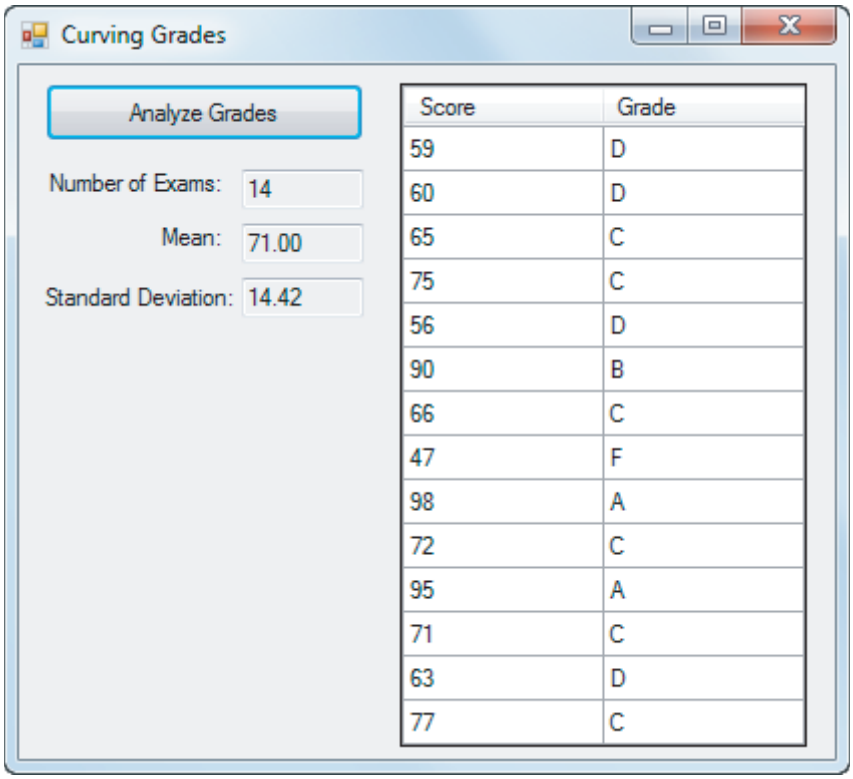

FIGURE 7.38 Output of Programming Project 2.

TABLE 7.17 English words and their French and German equivalents.

English	French	German	English	French	German
YES	OUI	JA	LARGE	GROS	GROSS
TABLE	TABLE	TISCH	NO	NON	NEIN
THE	LA	DEM	HAT	CHAPEAU	HUT
IS	EST	IST	PENCIL	CRAYON	BLEISTIFT
YELLOW	JAUNE	GELB	RED	ROUGE	ROT
FRIEND	AMI	FREUND	ON	SUR	AUF
SICK	MALADE	KRANK	AUTO	AUTO	AUTO
MY	MON	MEIN	OFTEN	SOUVENT	OFT

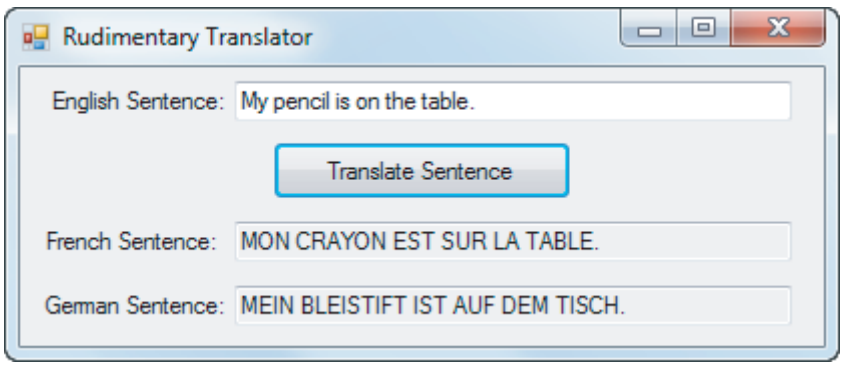

FIGURE 7.39 Possible outcome of Programming Project 3.

4. Table 7.18 shows the number of bachelor degrees conferred in 1981 and 2006 in certain fields of study. Tables 7.19 and 7.20 show the percentage change and a histogram of 2006 levels, respectively. Write a program that allows the user to display any one of these tables as an option and to quit as a fourth option. Table 7.18 is ordered alphabetically by field of study, Table 7.19 is ordered by decreasing percentages, and Table 7.20 is ordered by increasing number of degrees. ***Note:*** Chr(149) is a large dot.

TABLE 7.18 Bachelor degrees conferred in certain fields.

Field of Study	1981	2006
Business	200,521	318,042
Computer and info. science	15,121	47,480
Education	108,074	107,238
Engineering	63,642	67,045
Social sciences and history	100,513	161,485

Source: U.S. National Center of Education Statistics.

TABLE 7.19 Percentage change in bachelor degrees conferred.

Field of Study	% Change (1981–2006)
Computer and info. science	214.0 %
Social sciences and history	60.7 %
Business	58.6 %
Engineering	5.3 %
Education	−0.8 %

TABLE 7.20 Bachelor degrees conferred in 2006 in certain fields.

Field of Study	
Computer and info. science	••••• 47,480
Engineering	••••••• 67,045
Education	•••••••••••• 107,238
Social sciences and history	•••••••••••••••••• 161,485
Business	•••••••••••••••••••••••••••••••••••• 318,042

5. Each team in a six-team soccer league played each other team once. Table 7.21 shows the winners. Write a program to

(a) Place the team names in an array of structures that also holds the number of wins.

(b) Place the data from Table 7.21 in a two-dimensional array.

TABLE 7.21 Soccer league winners.

	Jazz	Jets	Owls	Rams	Cubs	Zips
Jazz	—	Jazz	Jazz	Rams	Cubs	Jazz
Jets	Jazz	—	Jets	Jets	Cubs	Zips
Owls	Jazz	Jets	—	Rams	Owls	Owls
Rams	Rams	Jets	Rams	—	Rams	Rams
Cubs	Cubs	Cubs	Owls	Rams	—	Cubs
Zips	Jazz	Zips	Owls	Rams	Cubs	—

(c) Place the number of games won by each team in the array of structures.

(d) Display a listing of the teams giving each team's name and number of games won. The list should be in decreasing order by the number of wins.

6. A poker hand can be stored in a two-dimensional array. The statement

```
Dim hand(3, 12) As Integer
```

declares an array with 52 elements, where the first subscript ranges over the four suits and the second subscript ranges over the thirteen denominations. A poker hand is specified by placing 1's in the elements corresponding to the cards in the hand. See Figure 7.40.

	A	2	3	4	5	6	7	8	9	10	J	Q	K
Club ♣	0	0	0	0	0	0	0	0	1	0	0	0	0
Diamond ♦	1	0	0	0	0	0	0	0	0	0	0	0	0
Heart ♥	1	0	0	0	0	0	0	0	0	0	0	1	0
Spade ♠	0	0	0	0	1	0	0	0	0	0	0	0	0

FIGURE 7.40 **Array for the poker hand A ♥ A ♦ 5 ♠ 9 ♣ Q ♥.**

Write a program that requests the five cards as input from the user, creates the related array, and passes the array to procedures to determine the type of the hand: flush (all cards have the same suit), straight (cards have consecutive denominations—ace can come either before 2 or after King), straight flush, four-of-a-kind, full house (three cards of one denomination, two cards of another denomination), three-of-a-kind, two pairs, one pair, or none of the above.

7. *Airline Reservations*. Write a reservation system for an airline flight. Assume the airplane has 10 rows with 4 seats in each row. Use a two-dimensional array of strings to maintain a seating chart. In addition, create an array to be used as a waiting list in case the plane is full. The waiting list should be "first come, first served"; that is, people who are added early to the list get priority over those added later. Allow the user the following three options:

(a) Add a passenger to the flight or waiting list.

1. Request the passenger's name.
2. Display a chart of the seats in the airplane in tabular form.
3. If seats are available, let the passenger choose a seat. Add the passenger to the seating chart.
4. If no seats are available, place the passenger on the waiting list.

(b) Remove a passenger from the flight.

1. Request the passenger's name.
2. Search the seating chart for the passenger's name and delete it.
3. If the waiting list is empty, update the array so the seat is available.
4. If the waiting list is not empty, remove the first person from the list, and give him or her the newly vacated seat.

(c) Quit.

8. The Game of Life was invented by John H. Conway to model some natural laws for birth, death, and survival. Consider a checkerboard consisting of an *n*-by-*n* array of squares. Each square can contain one individual (denoted by 1) or be empty (denoted by −). Figure 7.41(a) shows a 6-by-6 board with four of the squares occupied. The future of each individual depends on the number of his neighbors. After each period of time, called a *generation*, certain individuals will survive, others will die due to either loneliness or overcrowding, and new individuals will be born. Each nonborder square has eight neighboring squares. After each generation, the status of the squares changes as follows:

(a) An individual *survives* if there are two or three individuals in neighboring squares.

(b) An individual *dies* if he has more than three individuals or less than two in neighboring squares.

(c) A new individual *is born* into each empty square that has exactly three individuals as neighbors.

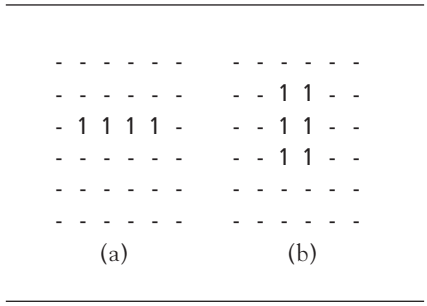

FIGURE 7.41 **Two generations.**

Figure 7.41(b) shows the status after one generation. Write a program to do the following:

1. Declare a two-dimensional array of size *n* by *n*, where *n* is input by the user, to hold the status of each square in the current generation. To specify the initial configuration, have the user input each row as a string of length *n*, and break the row into 1's or dashes with the Substring method.
2. Declare a two-dimensional array of size *n* by *n* to hold the status of each square in the next generation. Compute the status for each square and produce the display in Figure 7.41(b). ***Note:*** The generation changes all at once. Only current cells are used to determine which cells will contain individuals in the next generation.
3. Assign the next-generation values to the current generation and repeat as often as desired.
4. Display the individuals in each generation. (***Hint:*** The hardest part of the program is determining the number of neighbors a cell has. In general, you must check a 3-by-3 square around the cell in question. Exceptions must be made when the cell is on the edge of the array. Don't forget that a cell is not a neighbor of itself.)

(Test the program with the initial configuration shown in Figure 7.42. It is known as the figure-eight configuration and repeats after eight generations.)

9. Every book is identified by a ten-character International Standard Book Number (ISBN), which is usually printed on the back cover of the book. The first nine characters are digits and the last character is either a digit or the letter X (which stands for ten). Three examples of ISBNs are 0-13-030657-6, 0-32-108599-X, and 0-471-58719-2. The hyphens separate the characters into four blocks. The first block usually consists of a single digit and identifies the language (0 for English, 2 for French, 3 for German, etc.). The second block

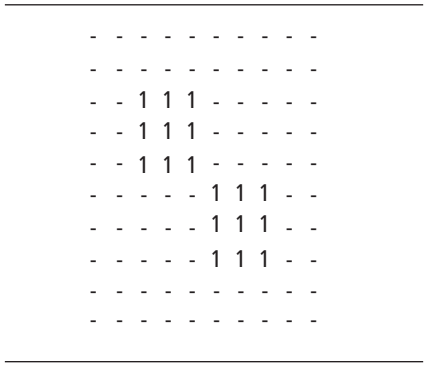

FIGURE 7.42 **The figure eight.**

identifies the publisher (for example, 13 for Prentice Hall, 32 for Addison-Wesley-Longman, and 471 for Wiley). The third block is the number the publisher has chosen for the book. The fourth block, which always consists of a single character called the *check digit*, is used to test for errors. Let's refer to the ten characters of the ISBN as d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, and d_{10}. The check digit is chosen so that the sum

$$10 \cdot d_1 + 9 \cdot d_2 + 8 \cdot d_3 + 7 \cdot d_4 + 6 \cdot d_5 + 5 \cdot d_6 + 4 \cdot d_7 + 3 \cdot d_8 + 2 \cdot d_9 + 1 \cdot d_{10} \quad (*)$$

is a multiple of 11. (***Note:*** A number is a multiple of 11 if it is exactly divisible by 11.) If the last character of the ISBN is an X, then in the sum (*), d_{10} is replaced with 10. For example, with the ISBN 0-32-108599-X, the sum would be

$$10 \cdot 0 + 9 \cdot 3 + 8 \cdot 2 + 7 \cdot 1 + 6 \cdot 0 + 5 \cdot 8 + 4 \cdot 5 + 3 \cdot 9 + 2 \cdot 9 + 1 \cdot 10 = 165$$

Since 165/11 is 15, the sum is a multiple of 11. This checking scheme will detect every single-digit and transposition-of-adjacent-digits error. That is, if while copying an IBSN number you miscopy a single character or transpose two adjacent characters, then the sum (*) will no longer be a multiple of 11.

(a) Write a program to accept an ISBN type number (including the hyphens) as input, calculate the sum (*), and tell if it is a valid ISBN. (***Hint:*** The number n is divisible by 11 if n Mod 11 is 0.) Before calculating the sum, the program should check that each of the first nine characters is a digit and that the last character is either a digit or an X.

(b) Write a program that begins with a valid ISBN (such as 0-13-030657-6) and then confirms that the checking scheme described above detects every single-digit and transposition-of-adjacent-digits error by testing every possible error. [***Hint:*** If d is a digit to be replaced, then the nine possibilities for the replacements are $(d + 1)$ Mod 10, $(d + 2)$ Mod 10, $(d + 3)$ Mod 10, ..., $(d + 9)$ Mod 10.]

10. *User-Operated Directory Assistance*. Have you ever tried to call someone at a place of business and been told to type in some letters of their name on your telephone's keypad in order to obtain their extension? Write a program to simulate this type of directory assistance. Suppose the names and telephone extensions of all the employees of a company are contained in the text file Employees.txt. Each set of three lines of the file has three pieces of information: last name, first and middle name(s), and telephone extension. (We have filled the file with the names of the U.S. presidents so that the names will be familiar.) The user should be asked to press buttons for the first three letters of the person's last name followed by the first letter of the first name. For instance, if the person's name were Gary Land, the

user would type in 5264. The number 5264 is referred to as the "push-button encoding" of the name. ***Note:*** People with different names can have the same push-button encoding—for instance, Herb James and Gary Land. After the user presses four keys on the keypad, the program should display the names and extensions of all the employees having the specified push-button encoding. See Fig. 7.43.

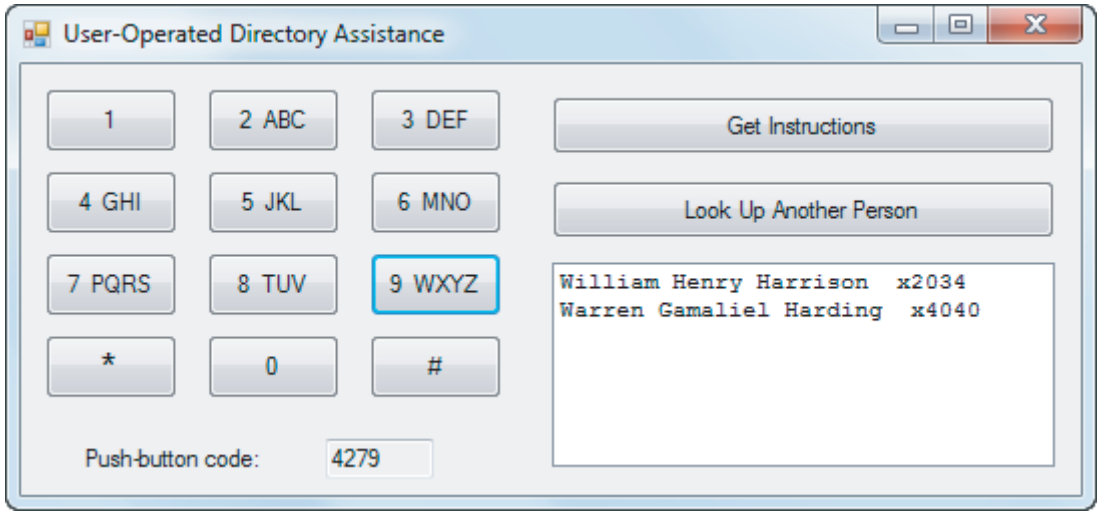

FIGURE 7.43 **Sample run of Programming Project 10.**

11. A fuel-economy study was carried out for five models of cars. Each car was driven 100 miles, and then the model of the car and the number of gallons used were placed in a line of the file Mileage.txt. Table 7.22 shows the data for the entries of the file. Write a program to display the models and their average miles per gallon in decreasing order with respect to mileage. The program should utilize an array of structures with upper bound 4, where each structure has three members. The first member should record the name of each model of car. The second member should record the number of test vehicles for each model. The third member should record the total number of gallons used by that model. [***Hint:*** Two Function procedures that are helpful have the headers `Function NumCars(ByVal make As String) As Integer` and `Function NumGals(ByVal make As String) As Double`. *NumCars* calculates the number of cars of the specified model in the table, and *NumGals* calculates the number of gallons used by the model. Both Function procedures are easily coded with LINQ queries.]

TABLE 7.22 **Gallons of gasoline used in 100 miles of driving.**

Model	Gal	Model	Gal	Model	Gal
Prius	2.1	Accord	4.1	Accord	4.3
Camry	4.1	Camry	3.8	Prius	2.3
Sebring	4.2	Camry	3.9	Camry	4.2
Mustang	5.3	Mustang	5.2	Accord	4.4