***3.2** (*Geometry: great circle distance*) The great circle distance is the distance between two points on the surface of a sphere. Let $(x1, y1)$ and $(x2, y2)$ be the geographical latitude and longitude of two points. The great circle distance between the two points can be computed using the following formula:

$$d = radius \times \arccos(\sin(x_1) \times \sin(x_2) + \cos(x_1) \times \cos(x_2) \times \cos(y_1 - y_2))$$

Write a program that prompts the user to enter the latitude and longitude of two points on the earth in degrees and displays its great circle distance. The average earth radius is 6,371.01 km. Note that you need to convert the degrees into radians using the `math.radians` function since the Python trigonometric functions use radians. The latitude and longitude degrees in the formula are for north and west. Use negative to indicate south and east degrees. Here is a sample run:

```
Enter point 1 (latitude and longitude) in degrees:
39.55, -116.25 ↵Enter
Enter point 2 (latitude and longitude) in degrees:
41.5, 87.37 ↵Enter
The distance between the two points is 10691.79183231593 km
```

***3.3** (*Geography: estimate areas*) Find the GPS locations for Atlanta, Georgia; Orlando, Florida; Savannah, Georgia; and Charlotte, North Carolina from www.gps-data-team.com/map/ and compute the estimated area enclosed by these four cities. (Hint: Use the formula in Programming Exercise 3.2 to compute the distance between two cities. Divide the polygon into two triangles and use the formula in Programming Exercise 2.14 to compute the area of a triangle.)

You may use the following coordinates to test your code.

```
lati_Charlotte, longi_Charlotte = 35.2270869, -80.8431267
lati_Savannh, longi_Savannah = 32.0835407, -81.0998342
lati_Orlando, longi_Orlando = 28.5383355, -81.3792365
lati_Atlanta, longi_Atlanta = 33.7489954, -84.3879824
```

The output woud be something like,

```
The area is 117863.34 square kilometers
>>>
```

***3.9** (*Financial application: payroll*) Write a program that reads the following information and prints a payroll statement:

Employee's name (e.g., Smith)
Number of hours worked in a week (e.g., 10)
Hourly pay rate (e.g., 9.75)
Federal tax withholding rate (e.g., 20%)
State tax withholding rate (e.g., 9%)

A sample run is shown below:

```
Enter employee's name: Smith Enter
Enter number of hours worked in a week: 10 Enter
Enter hourly pay rate: 9.75 Enter
Enter federal tax withholding rate: 0.20 Enter
Enter state tax withholding rate: 0.09 Enter

Employee Name: Smith
```

```
Hours Worked: 10.0
Pay Rate: $9.75
Gross Pay: $97.5
Deductions:
  Federal Withholding (20.0%): $19.5
  State Withholding (9.0%): $8.77
  Total Deduction: $28.27
Net Pay: $69.22
```

**4.21 (*Science: day of the week*) Zeller's congruence is an algorithm developed by Christian Zeller to calculate the day of the week. The formula is

$$h = \left(q + \left\lfloor \frac{26(m+1)}{10} \right\rfloor + k + \left\lfloor \frac{k}{4} \right\rfloor + \left\lfloor \frac{j}{4} \right\rfloor + 5j\right) \% 7$$

where

- `h` is the day of the week (0: Saturday, 1: Sunday, 2: Monday, 3: Tuesday, 4: Wednesday, 5: Thursday, 6: Friday).
- `q` is the day of the month.
- `m` is the month (3: March, 4: April, ..., 12: December). January and February are counted as months 13 and 14 of the previous year.
- `j` is the century (i.e., $\left\lfloor \frac{year}{100} \right\rfloor$).
- `k` is the year of the century (i.e., *year* % 100).

Write a program that prompts the user to enter a year, month, and day of the month, and then it displays the name of the day of the week. Here are some sample runs:

```
Enter year: (e.g., 2008): 2013 Enter
Enter month: 1-12: 1 Enter
Enter the day of the month: 1-31: 25 Enter
Day of the week is Friday
```

```
Enter year: (e.g., 2008): 2012 Enter
Enter month: 1-12: 5 Enter
Enter the day of the month: 1-31: 12 Enter
Day of the week is Saturday
```

(Hint: $\lfloor n \rfloor = n//1$ for a positive n. January and February are counted as `13` and `14` in the formula, so you need to convert the user input `1` to `13` and `2` to `14` for the month and change the year to the previous year.)

**4.28 (*Geometry: two rectangles*) Write a program that prompts the user to enter the center x-, y-coordinates, width, and height of two rectangles and determines whether the second rectangle is inside the first or overlaps with the first, as shown in Figure 4.10. Test your program to cover all cases.

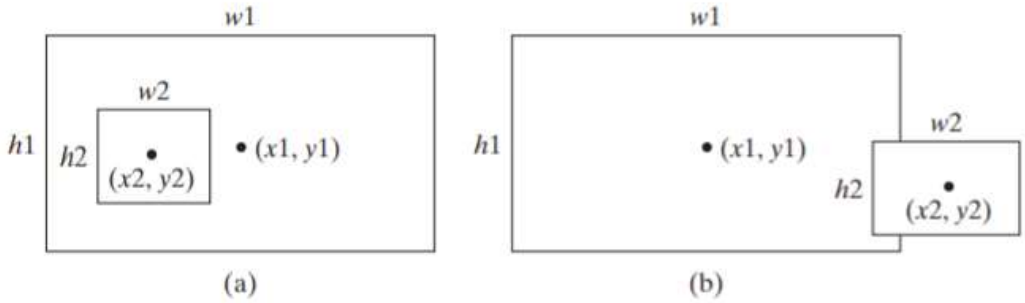

FIGURE 4.10 (a) A rectangle is inside another one. (b) A rectangle overlaps another one.

Here are some sample runs:

```
Enter r1's center x-, y-coordinates, width, and height:
  2.5, 4, 2.5, 43 Enter
Enter r2's center x-, y-coordinates, width, and height:
  1.5, 5, 0.5, 3 Enter
r2 is inside r1
```

```
Enter r1's center x-, y-coordinates, width, and height:
  1, 2, 3, 5.5 Enter
Enter r2's center x-, y-coordinates, width, and height:
  3, 4, 4.5, 5 Enter
r2 overlaps r1
```

```
Enter r1's center x-, y-coordinates, width, and height:
  1, 2, 3, 3 Enter
Enter r2's center x-, y-coordinates, width, and height:
  40, 45, 3, 2 Enter
r2 does not overlap r1
```

***4.33** (*Decimal to hex*) Write a program that prompts the user to enter an integer between 0 and 15 and displays its corresponding hex number. Here are some sample runs:

```
Enter a decimal value (0 to 15): 11 Enter
The hex value is B
```

```
Enter a decimal value (0 to 15): 5 Enter
The hex value is 5
```

```
Enter a decimal value (0 to 15): 31 Enter
Invalid input
```

***4.34** (*Hex to decimal*) Write a program that prompts the user to enter a hex character and displays its corresponding decimal integer. Here are some sample runs:

```
Enter a hex character: A Enter
The decimal value is 10
```

```
Enter a hex character: a Enter
The decimal value is 10
```

```
Enter a hex character: 5 Enter
The decimal value is 5
```

```
Enter a hex character: G Enter
Invalid input
```