CHAPTER 2: SOLUTIONS TO REVIEW EXERCISES

NSolution R2.1:

a. const int DAYS_PER_WEEK = 7;
b. int semester_days_left; // The user will need to enter information.
c. const double CENTIMETERS_PER_INCH = 2.54;
d. double tallest_height; // The user will need to enter information.

RSolution R2.2:
The value of `mystery` is equal to 0 after the statements are executed.

- In the first statement (line 1), `mystery` is initialized to a value of 1.
- In the calculation & assignment statement on line 2, `mystery` is still 1 on the right side of the statement, but the new value calculated, -1, is then assigned back to `mystery`.
- Finally, in the calculation & assignment on line 3, `mystery` on the right side of the statement is still -1, but is then incremented up to 0, which is assigned back to `mystery`.

Solution R2.3:
The variable `mystery` is being declared twice, first on line 1 and then again on line 3. A variable can only be initialized once. (If you remove the reserved word `int` on line 3, the statements will work just fine, and will update `mystery` to -3.)

Solution R2.4:
The mathematical expressions written in C++ are as follows:

```
s = s0 + v0 * t + (1 / 2) * g * t * t;

G = 4 * PI * PI * pow(a, 3) / (p * p * (m1 + m2));

FV = PV * pow((1 + (INT / 100)), YRS);
(note: INT is all caps, and not the reserved word int.)

c = sqrt(a * a + b * b - 2 * a * b * cos(gamma));
```

CSolution R2.5:
The C++ statements can be written as mathematical expressions as follows:

a. $$d_m = m\left(\frac{\sqrt{1+v/c}}{\sqrt{1-v/c}} - 1\right)$$

b. $$volume = \pi r^2 h$$

c. $$volume = \frac{4\pi r^3}{3}$$

d. $$z = \sqrt{x^2 + y^2}$$

Solution R2.6:
The values of the expressions are:

a. 6.25
b. 6
c. 12.5
d. -3
e. Because the `sqrt` function is expecting a floating point type of number (and not an integer) as input, you will likely get a warning when you try to execute this in a program. You can fix this by either:
    1. using a cast (see Special Topic 2.3) to convert n to a double before executing the program, or
    2. multiplying n by 1.0 to convert its type to double for use within the sqrt() function, or
    3. dividing n by 1.0 to convert it to double for use within the sqrt() function.

Solution R2.7:
The values of the expressions are:

a. 10
b. el
c. l   // this is the first lower case L in the word “Hello”
              // note that 5/2 is 2, because both are type int.
d. HelloWorld
e. WorldHello

Solution R2.8:
Find at least 5 compile-time errors in the program (line numbers added for reference):

```
1)   #include iostream
2)
3)   int main();
4)   {
5)      cout << "Please enter two numbers:"
6)      cin << x, y;
7)      cout << "The sum of << x << "and" << y
8)           << "is: " x + y << endl;
9)      return;
10)  }
```

Compile-Time Errors:

1. The term `iostream` should be either <`iostream`> or “`iostream`”.
2. There is no `using namespace std` line between the preprocessor directive and the `int main()` line.
3. There should not be a semicolon at the end of Line 3.
4. There is a missing semicolon at the end of Line 5.
5. The << operator should be a >> operator.
6. The comma in Line 6 should be a >> operator.
7. There is a missing double quote on Line 7 (after `The sum of`, before the << operator).
8. There is a missing << operator on Line 8 (after the literal, and before `x + y`).
9. The return statement on Line 9 should return a value.

Solution R2.9:
Find at least 4 run-time errors in the program (line numbers added for reference):

```
1)   #include <iostream>
2)
3)   using namespace std;
4)
5)   int main()
6)   {
7)      int total;
8)      int x1;
9)      cout << "Please enter a number:";
10)     cin >> x1;
11)     total = total + x1;
12)     cout << "Please enter another number:";
13)     int x2;
14)     cin >> x2;
15)     total = total + x1;
16)     double average = total / 2;
17)     cout << "The average of the two numbers is "
18)        << average << "endl";
19)     return 0;
20)  }
```

Run-time errors (also known as Logic Errors):

1. The prompts do not specify integers be entered; if a number with a decimal part is entered, the decimal part will be truncated.
2. The `total` variable should be initialized to zero before being used on Line 11.
3. The variable `x1` was mistakenly used instead of `x2` on Line 15.
4. Since the variable total was defined to be an integer, the division on Line 16 will be treated as an integer division, and digits after the decimal point in the result will be truncated. So, total should have been declared as type double.
5. There should not be double quotes around `endl` on Line 18.

Solution R2.10:
The values 2 and 2.0 are both number literals, and “2” and “2.0” are string literals.

The value 2 is considered to be an integer number, because it represents a whole number and has no decimal part. The value 2.0 is considered to be a floating point number, as it has a decimal portion.

When you add double quotes to the numbers, as with “2” and “2.0”, you create string literals. String literals are interpreted as a sequence of characters and not as numbers. The string literal “2” is seen by the computer as a single character, and the string literal “2.0” is seen as a sequence of three characters.

Solution R2.11:
In Part (a) of this problem, the variable x is declared and initialized to the integer value 2. Therefore, on the second line, the variable y is declared and assigned the result of executing the ***sum*** of x plus x, which is 4.

In Part (b), the variable s is declared and assigned the string literal “2”, which is a character, not a number. Therefore, on the second line of Part (b), the variable t is declared and assigned the result of the concatenation of “2” and “2”, which is the character sequence “22”.

Solution R2.12:
Pseudocode for a program that rearranges the letters of a word.

```
Prompt user to enter a word.
Read a word from the keyboard.
Set first = the substring starting at position 0 in word and 1 character long.
Set last = the substring starting at position (word.length – 1) in the word
   string and 1 character long.
Set rest = the substring starting at position 1 in word and ending at position
   (word.length – 2).
Print first + “ ” + last + “ ” + rest.
```

Solution R2.13:
Pseudocode for a program that prints a person’s monogram.

```
Prompt user to enter a first name.
Read the name from the keyboard.
Set first_name = name read from keyboard.
Prompt user to enter a middle name.
Read the name from the keyboard.
Set middle_name = name read from keyboard.
Prompt user to enter a last name.
Read the name from the keyboard.
Set last_name = name read from keyboard.
Set monogram = first letter of first_name + first letter of middle_name
       + first letter of last_name.
Print monogram.
```

Note that the first letter is found using the substring function, starting at position 0 and 1 character long.

RSolution R2.14:
Pseudocode for a program that prints the first and last digits of a number.

```
Prompt user to enter a number.
Set num = number read from keyboard.
Set last = num % 10.
Set first = integer portion of (num / (pow(10, integer portion of log10(m)))).
Set display_num =10 * first + last.
Print display_num.
```

RSolution R2.15: Pseudocode for How To 2.1 program giving dimes and nickels as well as quarters.

```
amount due = 100 x bill value - item price in pennies
quarters = amount due / 25 (without remainder)
amount due = amount due – (quarters * 25)
dimes = amount due / 10 (without remainder)
amount due = amount due – (dimes * 10)
nickels = amount due / 5
```

Solution R2.16:
First, compute the total volume by hand using the formula provided. Let's assume the following values for the three sections:

- Conic Section 1 (top of bottle) has first radius ($r1_1$) 0.5 inches, second radius ($r2_1$) 0.75 inches, and height (h_1) 1.5 inches.
- Conic Section 2 (middle) has first radius ($r1_2$) 0.75 inches, second radius ($r2_2$) 1.5 inches, and height (h_2) 0.5 inches.
- Conic Section 3 (bottom of bottle) has first radius ($r1_3$) 1.5 inches, second radius ($r2_3$) 0.75 inches, and height (h_3) 6.0 inches.

So, the total volume of the bottle, V_t, will equal the volume of conic section 1 (V_1) plus the volume of conic section 2 (V_2) plus the volume of the conic section 3 (V_3):

$$V_t = V_1 + V_2 + V_3$$

Using the equation given in the problem, we calculate the three volumes to be (rounded to two decimal places):

V_1 = 1.87 cubic inches
V_2 = 2.06 cubic inches
V_3 = 24.74 cubic inches

So, the total volume for the cocktail shaker in our example is:

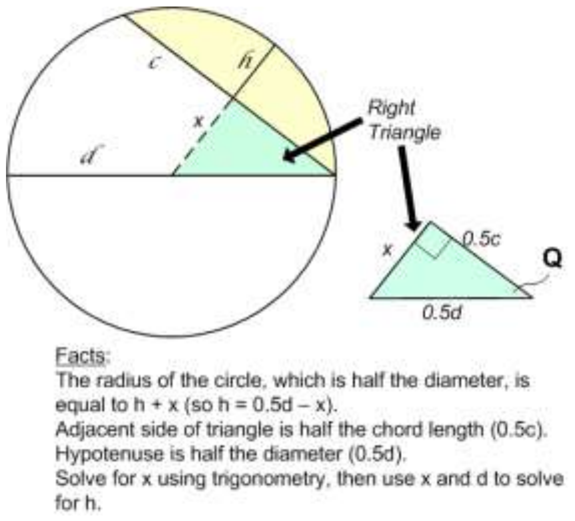
$V_t = V_1 + V_2 + V_3 = 28.67$ cubic inches

The algorithm we use is very similar to the calculation we did above:

```
volume 1 = $\pi \times (r1_1 \times r1_1 + r1_1 \times r2_1 + r2_1 \times r2_1) \times h1 / 3$
volume 2 = $\pi \times (r1_2 \times r1_2 + r1_2 \times r2_2 + r2_2 \times r2_2) \times h2 / 3$
volume 3 = $\pi \times (r1_3 \times r1_3 + r1_3 \times r2_3 + r2_3 \times r2_3) \times h3 / 3$
total volume = volume 1 + volume 2 + volume 3
```

RSolution R2.17:

Because we are given the diameter of the circle, d, and not the height of the circular sector, h, we need to develop an algorithm to find h based on the value of d.



Facts:
The radius of the circle, which is half the diameter, is equal to h + x (so h = 0.5d – x).
Adjacent side of triangle is half the chord length (0.5c).
Hypotenuse is half the diameter (0.5d).
Solve for x using trigonometry, then use x and d to solve for h.

If you look at the figure on the right, you see that a right triangle can be made bounded by the chord, the diameter of the circle, and the radius of the circle drawn through h.

Because we know the length of the radius (half the diameter), we can set up an equation to find the value of h:

$$h = 0.5d - x$$

where x is one side of the right triangle. We can solve for x by using trigonometry on the right triangle, and then use that value to solve for the value of h, which we need to compute the area of the circular sector (the oddly-shaped piece of pie). In the right triangle, the side labeled x is considered to be the “opposite” side. We know that the hypotenuse of the right triangle is half the diameter, or 0.5d, and the “adjacent” side of the right triangle is half the chord length, or 0.5c.
Therefore, we can use the following formulas for a right triangle to determine what we need for the problem:

```
sin Q = opposite/hypotenuse      cos Q = adjacent/hypotenuse
```

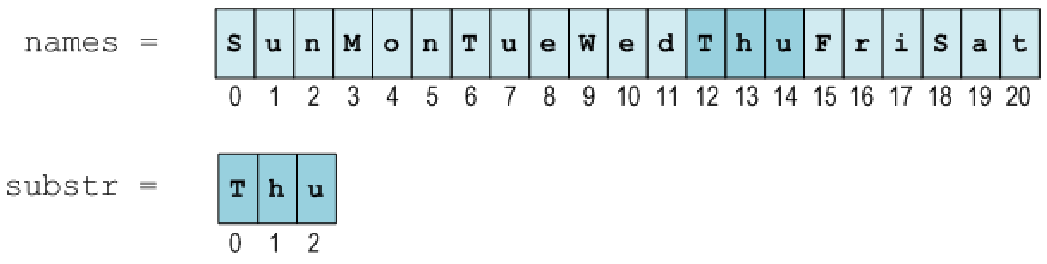
Here is the general algorithm (see figure for definition of the variables):

```
Ask the user to enter the diameter of the circle, store the value in d.
Ask the user to enter the length of the chord, store the value in c.
Compute the angle $\theta$ with formula $\theta = \cos^{-1}(0.5c / 0.5d)$.
Compute the length of x with the formula $x = 0.5d * \sin\theta$.
Determine h using the formula $h = 0.5d - x$.
Calculate the area of the pie piece, A, using the formula.
A = 2.0 / 3 * c * h + pow(h, 3) / (2 * c).
```

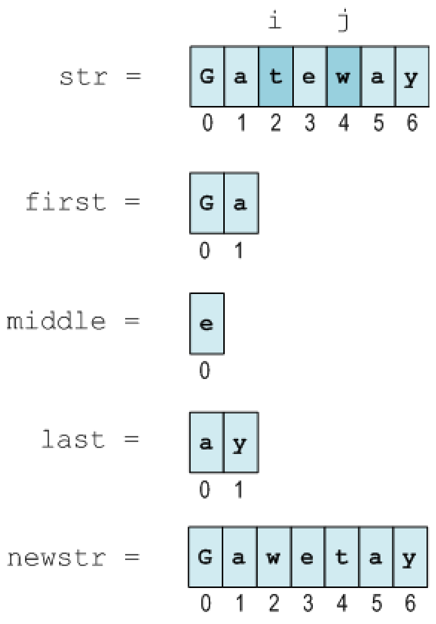
We can use this algorithm to solve for the specific example of a pie with diameter 12 inches and a chord length of 10 inches:

> We know that d = 12, c = 10
> Therefore, $\theta = \cos^{-1}(0.5c/0.5d) = \cos^{-1}(5/6) = 33.56$ degrees
> $x = \sin\theta * 0.5d = \sin 33.56 * 6 = 3.32$ inches
> $h = 0.5d - x = 6 - 3.32 = 2.68$ inches
> $A = 2/3 * c * h + h^3/(2*c) = 2/3 * 10 * 2.68 + 2.68^3 / (2 * 10) = 18.83$ square inches

Solution R2.18:
Figures showing the creation of a new substring.

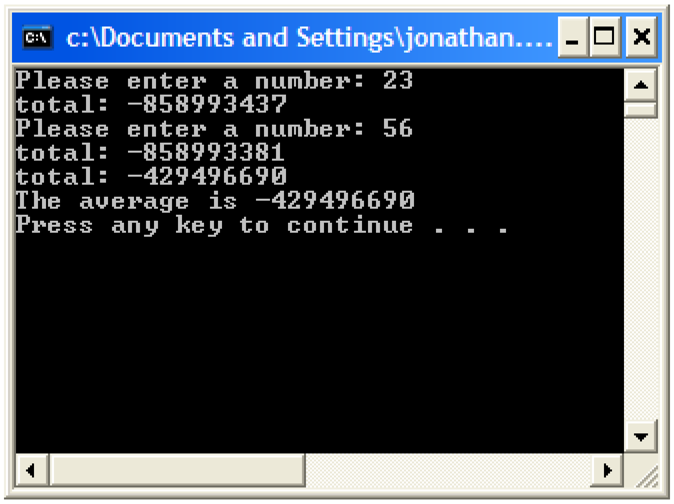


Solution R2.19:
Figures showing the creation of a new string by rearranging parts of the original string.



Solution R2.20:
When the program runs, it first prompts the user to enter a number. It stores this number in `x1` and then adds `x1` to the `total` variable. When the program hits the line where the `total` variable is first used, an error message is displayed stating that the `total` variable is being used before it was initialized.

If the debugger allows you to continue, the value that is displayed for total (the first `cout` line) is based on whatever random value was in the `total` variable when the program started. (See figure below for example output.)



```
Please enter a number: 23
total: -858993437
Please enter a number: 56
total: -858993381
total: -429496690
The average is -429496690
Press any key to continue . . .
```

The second error occurs because the integer variable `total` is used to compute and hold the average. This is misleading to the programmer as well as the user of the program (because a variable called total is holding the sum). Also, the `total` variable should not be an integer, since the average will very likely not be an integer number (and storing a non-integer result in an integer variable will result in loss of information).

The first two errors in the program can be easily fixed by declaring the `total` variable as `double` and initializing it to 0.0. The third error is fixed by declaring a second `double` variable called `average` and using that instead of the variable `total` when dividing by 2.

NSolution R2.21:
For a string named str:

1) Keep track of the length of str with:
   int length_of_str = str.length();
2) Find the first character with:
   int first_character = str.substr(0,1);
3) Find the last character with:
   Int last_character = str.substr(length_of_str – 1, 1);
4) To remove the first character:
   int without_first_character = str.substr(1,length_of_str – 1);
5) To remove the last character:
   Int without_last_character = str.substr(0, length_of_str – 1);

NSolution R2.22:

a. Exact.
b. Exact
c. Overflow
d. Overflow

Solution R2.23:
The program that prints the values can be written as follows (use the `endl` character to put each value on a line by itself):

```
#include <iostream>
using namespace std;
int main()
{
	cout << 3 * 1000 * 1000 * 1000 << endl;
	cout << 3.0 * 1000 * 1000 * 1000 << endl;
	return 0;
}
```

The output on my computer when this program was executed was:

```
	-1294967296
	3e+009
```

The first value displayed is a useless value without any real meaning. It is created because the calculation in the first `cout` line produced a result that was an integer value (because all of the number literals used in the formula were integers), and the intended result of 3,000,000,000 was too large to fit in the valid range of the integer data type as defined on my computer. The result is said to “overflow”, and only a truncated portion of the result is displayed.

The second value displayed is correct, because the first number literal used in the formula was a floating-point value. Because one of the number literals was a floating-point value, the output will also be floating-point, and the valid range of floating-point values is large enough to hold the result.

The second value is displayed in exponential notation. This notation is equivalent to saying 3 X 10^9, or 3,000,000,000.

Solution R2.24:
Recommendations for using variables and constants:

- Use descriptive variables names (for example, use **ounces_per_can** instead of simply **x** or **opc**).
- Initialize variables before they are used.
- The names of constants should always be UPPERCASE to distinguish them from regular variables.
- Use a constant to help explain the meaning of numeric values (for example, using **PENNIES_PER_DOLLAR** instead of simply using the number 100).
- Constants should be declared at the top of the function in which they are used.
- Variables should only be declared just before they are needed.
- Use comments near variables and constants to provide an explanation of what the value is or how it is used.