14.1 Example: Health data

Calculating user's age in days

The section presents an example program that computes various health related data based on a user's age using incremental development. ***Incremental development*** is the process of writing, compiling, and testing a small amount of code, then writing, compiling, and testing a small amount more (an incremental amount), and so on.

The initial program below calculates a user's age in days based on the user's age in years. The assignment statement `userAgeDays = userAgeYears * 365;` assigns userAgeDays with the product of the user's age and 365, which does not take into account leap years.

Figure 14.1.1: Health data: Calculating user's age in days.

```
#include <stdio.h>

int main(void) {
   int userAgeYears;
   int userAgeDays;

   printf("Enter your age in years: ");
   scanf("%d", &userAgeYears);

   userAgeDays = userAgeYears * 365;

   printf("You are %d days old.\n", userAgeDays);

   return 0;
}
```

```
Enter your age in years: 19
You are 6935 days old.
```

PARTICIPATION ACTIVITY 14.1.1: Calculating user age in days.

1) Which variable is used for the user's age in years?

Check Show answer

2) If the user enters 10, what will userAgeYears be assigned?

Check Show answer

3) If the user enters 10, what is userAgeDays assigned?

Check Show answer

Considering leap years and calculating age in minutes

The program below extends the previous program by accounting for leap years when calculating the user's age in days. Since each leap year has one extra day, the statement `userAgeDays = userAgeDays + (userAgeYears / 4)` adds the number of leap years to userAgeDays. Note that the parentheses are not needed but are used to make the statement easier to read.

The program also computes and outputs the user's age in minutes.

Figure 14.1.2: Health data: Calculating user's age in days and minutes.

```
#include <stdio.h>

int main(void) {
   int userAgeYears;
   int userAgeDays;
   int userAgeMinutes;

   printf("Enter your age in years: ");
   scanf("%d", &userAgeYears);

   userAgeDays = userAgeYears * 365;                // Calculate days without leap years
   userAgeDays = userAgeDays + (userAgeYears / 4); // Add days for leap years

   printf("You are %d days old.\n", userAgeDays);

   userAgeMinutes = userAgeDays * 24 * 60;         // 24 hours/day, 60 minutes/hour
   printf("You are %d minutes old.\n", userAgeMinutes);

   return 0;
}
```

```
Enter your age in years: 19
You are 6939 days old.
You are 9992160 minutes old.
```

PARTICIPATION ACTIVITY 14.1.2: Calculating user age in days.

1) The expression `(userAgeYears /`

`4)` assumes a leap year occurs every four year?

- ○ True
- ○ False

2) The statement `userAgeDays = userAgeDays + (userAgeYears / 4);` requires parentheses to evaluate correctly.

- ○ True
- ○ False

3) If the user enters 20, what is userAgeDays after the first assignment statement?

- ○ 7300
- ○ 7305

4) If the user enters 20, what is userAgeDays after the second assignment statement?

- ○ 7300
- ○ 7305

Estimating total heartbeats in user's lifetime

The program is incrementally extended again to calculate the approximate number of times the user's heart has beat in his/her lifetime using an average heart rate of 72 beats per minutes.

Figure 14.1.3: Health data: Calculating total heartbeats lifetime.

```
#include <stdio.h>

int main(void) {
   int userAgeYears;
   int userAgeDays;
   int userAgeMinutes;
   int totalHeartbeats;
   int avgBeatsPerMinute = 72;

   printf("Enter your age in years: ");
   scanf("%d", &userAgeYears);

   userAgeDays = userAgeYears * 365;              // Calculate days without leap years
   userAgeDays = userAgeDays + (userAgeYears / 4); // Add days for leap years

   printf("You are %d days old.\n", userAgeDays);

   userAgeMinutes = userAgeDays * 24 * 60;        // 24 hours/day, 60 minutes/hour
   printf("You are %d minutes old.\n", userAgeMinutes);

   totalHeartbeats = userAgeMinutes * avgBeatsPerMinute;
   printf("Your heart has beat %d times.\n", totalHeartbeats);

   return 0;
}
```

```
Enter your age in years: 19
You are 6939 days old.
You are 9992160 minutes old.
Your heart has beat 719435520 times.
```

PARTICIPATION ACTIVITY 14.1.3: Calculating user's heartbeats.

1) Which variable is initialized when declared?

- ○ userAgeYears
- ○ totalHeartbeats
- ○ avgBeatsPerMinute

2) If the user enters 10, what value is held in totalHeartbeats after the statement `userAgeDays = userAgeYears * 365;`

- ○ 3650
- ○ 5258880
- ○ Unknown

14.2 Scientific notation for floating-point literals

Scientific notation is useful for representing floating-point numbers that are much greater than or much less than 0, such as 6.02 x 10^{23}. A floating-point literal using ***scientific notation*** is written using an e preceding the power-of-10 exponent, as in 6.02e23 to represent 6.02 x 10^{23}. The e stands for exponent. Likewise, 0.001 is 1 x 10^{-3} and can be written as 1.0e-3. For a floating-point literal, good practice is to make the leading digit non-zero.

Figure 14.2.1: Calculating atoms of gold.

```
#include <stdio.h>

int main(void) {
   double avogadrosNumber = 6.02e23; // Approximation of atoms per mole
   double gramsPerMoleGold = 196.9665;
   double gramsGold;
   double atomsGold;

   printf("Enter grams of gold: ");
   scanf("%lf", &gramsGold);

   atomsGold = gramsGold / gramsPerMoleGold * avogadrosNumber;

   printf("%lf grams of gold contains ", gramsGold);
   printf("%lf atoms\n", atomsGold);

   return 0;
}
```

```
Enter grams of gold: 4.5
4.500000 grams of gold contains 13753607847019670405120.000000 atoms
```

PARTICIPATION ACTIVITY 14.2.1: Scientific notation.

1) Type 1.0e-4 as a floating-point literal with a single digit before and four digits after the decimal point. Note: Do not use scientific notation.

Check Show answer

2) Type 7.2e-4 as a floating-point literal with a single digit before and five digits

after the decimal point. Note: Do not use scientific notation.

Check Show answer

3) Type 540,000,000 as a floating-point literal using scientific notation with a single digit before and after the decimal point.

Check Show answer

4) Type 0.000001 as a floating-point literal using scientific notation with a single digit before and after the decimal point.

Check Show answer

5) Type 623.596 as a floating-point literal using scientific notation with a single digit before and five digits after the decimal point.

Check Show answer

CHALLENGE ACTIVITY 14.2.1: Acceleration of gravity.

Compute the acceleration of gravity for a given distance from the earth's center, distCenter, assigning the result to accelGravity. The expression for the acceleration of gravity is: *(G * M) / (d²)*, where G is the gravitational constant 6.673×10^{-11}, M is the mass of the earth 5.98×10^{24} (in kg) and d is the distance in meters from the earth's center (stored in variable distCenter).

```
#include <stdio.h>

int main(void) {
   double G = 6.673e-11;
   double M = 5.98e24;
```

```
    double accelGravity;
    double distCenter;

    distCenter = 6.38e6;

    /* Your solution goes here  */

    printf("accelGravity: %lf\n", accelGravity);

    return 0;
}
```

Run

14.3 Detecting ranges (general)

Range detection using if-elseif-else

An if-elseif-else structure can elegantly detect number ranges, such as under 6, 6 - 7, 8 - 9, 10 - 11, and 12 and up, with each branch performing a different action for each range. Each expression only needs to indicate the upper range part; if execution reaches an expression, the lower range part is implicit from the previous expressions being false.

PARTICIPATION ACTIVITY 14.3.1: An if-elseif-else structure can elegantly detect ranges.

Animation captions:

1. Kids of various ages may wish to play soccer. A soccer club may not have teams for kids 5 anc under.
2. One level of teams is listed as "Under 8" (or just U8), which is understood to mean just 7 or 6, but not 5 or younger.
3. Likewise, U10 means 9 and 8, and U12 means 11 and 10. No teams exist for ages 12 and over.
4. An if-elseif-else structure can elegantly capture such ranges. When an expression is checked, one knows that all the previous expressions were false, thus defining the low range end.

PARTICIPATION ACTIVITY 14.3.2: Using if-elseif-else to detect increasing ranges.

Indicate the range corresponding to each branch. x is a non-negative integer.

30+ 20 - 29 10 - 19 0 - 9

If x < 10 : Branch 1

Else If x < 20 : Branch 2

Else if x < 30 : Branch 3

Else : Branch 4

Reset

PARTICIPATION ACTIVITY 14.3.3: More ranges with if-elseif-else.

Indicate the range detected by the expression, assuming each question continues a single if-elseif-else structure. x is an integer. Type ranges as: 25 - 29

1) If x > 100 : Branch 1

```
 - infinity
```

Check Show answer

2) Else If x > 50 : Branch 2

Check Show answer

3) Else

```
-infinity -
```

Check Show answer

4) Is this a reasonable if-elseif-else structure? Type yes or no.

If x < 100: Branch 1
Else If x < 200: Branch 2
Else If x < 150: Branch 3
Else: Branch 4

Check Show answer

14.4 Detecting ranges with if-else statements

Programmers commonly use the sequential nature of the multi-branch if-else arrangement to detect ranges of numbers. In the following example, the second branch expression is only reached if the first expression is false. So the second branch is taken if userAge < 16 *is false* (so 16 or greater) AND userAge is < 25, meaning userAge is between 16 - 24 (inclusive).

Figure 14.4.1: Using sequential nature of multi-branch if-else for ranges: Insurance prices.

```
#include <stdio.h>

int main(void) {
   int userAge;
   int insurancePrice;

   printf("Enter your age: ");
   scanf("%d", &userAge);

   if (userAge < 16) {                // Age 15 and under
      printf("Too young.\n");
      insurancePrice = 0;
   }
   else if (userAge < 25) {           // Age 16..24
      insurancePrice = 4800;
   }
   else if (userAge < 40) {           // Age 25..39
      insurancePrice = 2350;
   }
   else {                              // Age 40 and up
      insurancePrice = 2100;
   }

   printf("Annual price: $%d\n", insurancePrice);

   return 0;
}
```

```
Enter your age: 19
Annual price: $4800

...

Enter your age: 27
Annual price: $2350

...

Enter your age: 15
Too young.
Annual price: $0

...

Enter your age: 129
Annual price: $2100
```

Source: cardsdirect.com, 2017

PARTICIPATION ACTIVITY 14.4.1: Ranges and multi-branch if-else.

Type the range for each branch. Type ranges as: 25 - 29, or type 30+ for all numbers 30 and larger.

```
if (numSales < 10) {
   ...
}
else if (numSales < 20) {  // 2nd branch range: ______
   ...
}
else if (numSales < 30) {  // 3rd branch range: ______
   ...
}
else {                     // 4th branch range: ______
   ...
}
```

1) 2nd branch range:

Check Show answer

2) 3rd branch range:

Check Show answer

3) 4th branch range:

Check Show answer

4) What is the range for the last branch below?

```
if (numItems < 0) {
   ...
}
else if (numItems > 100) {
   ...
}
else {  // Range: _______
   ...
}
```

Check Show answer

PARTICIPATION ACTIVITY

14.4.2: Complete the multi-branch if-else.

1) Second branch: userNum is less than 200

```
if (userNum < 100 ) {
   ...
}
else if (                    )
{
   ...
}
else { // userNum >= 200
   ...
}
```

Check Show answer

2) Second branch: userNum is positive (non-zero)

```
if (userNum < 0 ) {
   ...
}
                      {
   ...
}
else { // userNum is 0
   ...
}
```

Check Show answer

3) Second branch: userNum is greater than 105

```
if (userNum < 100 ) {

   ...

}

                 {

   ...

}

else { // userNum is between

      // 100 and 105

   ...

}
```

Check Show answer

4) If the final else branch executes, what must userNum have been? Type "unknown" if appropriate.

```
if (userNum <= 9) {
   ...
}
else if (userNum >= 11) {
   ...
}
else {
   ... // userNum if this executes?
}
```

Check Show answer

5) Which branch will execute? Valid answers: 1, 2, 3, or none.

```
userNum = 555;

if (userNum < 0) {
   ... // Branch 1
}
else if (userNum == 0) {
   ... // Branch 2
}
else if (userNum < 100) {
   ... // Branch 3
}
```

Check Show answer

CHALLENGE ACTIVITY 14.4.1: Multi-branch if-else statement: Print century.

Write an if-else statement with multiple branches. If givenYear is 2101 or greater, print "Distant future" (without quotes). Else, if givenYear is 2001 or greater (2001-2100), print "21st century". Else, if givenYear is 1901 or greater (1901-2000), print "20th century". Else (1900 or earlier), print "Long ago". Do NOT end with newline.

```
#include <stdio.h>

int main(void) {
   int givenYear;

   givenYear = 1776;

   /* Your solution goes here  */

   return 0;
}
```

Run

14.5 Order of evaluation

Precedence rules

The order in which operators are evaluated in an expression are known as ***precedence rules***. Arithmetic, logical, and relational operators are evaluated in the order shown below.

Table 14.5.1: Precedence rules for arithmetic, logical, and relational operators.

Operator/Convention	Description	Explanation
( )	Items within parentheses are evaluated first	In `(a * (b + c)) - d`, the + is evaluated first, then *, then -.
!	! (logical NOT) is next	`! x || y` is evaluated as `(!x) || y`
* / % + -	Arithmetic operators (using their precedence rules; see earlier section)	`z - 45 * y < 53` evaluates * first, then -, then <.
`< <= > >=`	Relational operators	`x < 2 || x >= 10` is evaluated as `(x < 2) || (x >= 10)` because < and >= have precedence over ||.
== !=	Equality and inequality operators	`x == 0 && x >= 10` is evaluated as `(x == 0) && (x >= 10)` because < and >= have precedence over &&. == and != have the same precedence and are evaluated left to right.
&&	Logical AND	`x == 5 || y == 10 && z != 10` is evaluated as `(x == 5) || ((y == 10) && (z != 10))` because && has precedence over ||.
||	Logical OR	

PARTICIPATION ACTIVITY 14.5.1: Applying the precedence rules to an expression can be thought of as a 'tree'.

Animation captions:

1. Expressions like x + 1 > y * z || z == 3 are evaluated using precedence rules. Among +, >, *, ||, an ==, the * comes first.
2. Next comes +, then ==, and finally ||.
3. The expression is actually treated like a "tree", evaluated from the bottom upwards.
4. If x is 7, y is 6, and z is 3, then y * z is 18. Next, x + 1 is 8. Next, 8 > 18 is false. Next, z == 3 is tru Finally, false || true is true.

PARTICIPATION ACTIVITY 14.5.2: Order of evaluation.

To teach precedence rules, these questions intentionally omit parentheses; good style would use parentheses to make order of evaluation explicit.

1) Which operator is evaluated first?
! y && x

- &&
- !

2) Which operator is evaluated first?
w + 3 > x - y * z

- +
- -
- >
- *

3) In what order are the operators evaluated?
w + 3 != y - 1 && x

- +, !=, -, &&
- +, -, &&, !=
- +, -, !=, &&

4) To what does this expression evaluate, given int x = 4, int y = 7.
x == 3 || x + 1 > y

- true
- false

Common error: Missing parentheses

A common error is to write an expression that is evaluated in a different order than expected. Good practice is to use parentheses in expressions to make the intended order of evaluation explicit. Several examples are below.

PARTICIPATION ACTIVITY 14.5.3: Common errors in expressions.

1) Does `! x == 3` evaluate as `!(x == 3)`?

- ○ Yes
- ○ No

2) Does `w + x  == y + z` evaluate as `(w + x) == (y + z)`?

- ○ Yes
- ○ No

3) Does `w && x == y && z` evaluate as `(w && x) == (y && z)`?

- ○ Yes
- ○ No

PARTICIPATION ACTIVITY 14.5.4: Order of evaluation.

Which illustrates the actual order of evaluation via parentheses?

1) `! green == red`

- ○ (!green) == red
- ○ !(green == red)
- ○ (!green =)= red

2) `bats < birds || birds < insects`

- ○ ((bats < birds) || birds) < insects
- ○ bats < (birds || birds) < insects
- ○ (bats < birds) || (birds < insects)

3) `! (bats < birds) || (birds < insects)`

- ○

! ((bats < birds) || (birds < insects))

- ○ (! (bats < birds)) || (birds < insects)
- ○ ((!bats) < birds) || (birds < insects)

4) `(num1 == 9) || (num2 == 0) && (num3 == 0)`

- ○ (num1 == 9) || ((num2 == 0) && (num3 == 0))
- ○ ((num1 == 9) || (num2 == 0)) && (num3 == 0)
- ○ (num1 == 9) || (num2 == (0 && num3) == 0)

Common error: Math expression for range

A common error often made by new programmers is to write expressions like `(16 < age < 25)`, as one might see in mathematics.

The meaning, however, almost certainly is not what the programmer intended. Suppose age is presently 28. The expression is evaluated left-to-right, so evaluation of `16 < age` yields true. Next, the expression `true < 25` is evaluated; clearly not the programmer's intent. However, true is actually 1, and evaluating `1 < 25` will yield true. Thus, for any age greater than 16, the above expression evaluates to true, even for ages greater than 25.

Thus, `16 < age < 25` is actually the same as `(16 < age) < 25`, which evaluates to `(true) < 25` for any age over 16, which is the same as `(1) < 25`, which evaluates to true. The correct way to do such a comparison is: `(age > 16) && (age < 25)`.

PARTICIPATION ACTIVITY 14.5.5: Expression for detecting a range.

1) A programmer erroneously wrote an expression as: 0 < x < 10. Rewrite the expression using logical AND. Use parentheses.

`(0 < x)` ____

Check Show answer

Common error: Bitwise rather than logical operators

Logical AND is && and not just &, and logical OR is || and not just |. & and | represent ***bitwise operators***, which perform AND or OR on corresponding individual bits of the operands.

A common error is to use a bitwise operator instead of a logical operator, typing & instead of &&, or typing | instead of ||. A bitwise operator may yield different behavior than expected.

PARTICIPATION ACTIVITY 14.5.6: Bitwise vs. logical operators.

Indicate if the expression correctly uses logical operators.

1) (x > 5) & (y > 3) & (z != 0)

- Yes
- No

2) (x == 0) || (y == 0) | (z == 0)

- Yes
- No

3) ((x == y) && (y == z)) || (w == 0)

- Yes
- No

14.6 Example: Toll calculation

Calculating toll based on time of day

The section presents an example program that calculates the toll amount for travel along a toll road or toll lane. The toll amount is based on the time of time, day of the week, and number of persons in the vehicle.

The initial version of the program calculates the toll amount for travel on a weekday based upon the toll schedule below. The table lists times in both am/pm format and 24-hour format.

Table 14.6.1: Weekday toll schedule.

Time (am/pm)	Time (24 hour)	Toll amount

Before 6:00 am	Before 6:00	1.55
6:00 am to 9:59 am	6:00 to 9:59	4.65
10:00 am to 5:59 pm	10:00 to 17:59	2.35
6:00 pm and after	18:00 and after	1.55

The program gets the time of travel from the user using 24 hours format, and uses the hour to determine the toll amount. A multi-branch if-else statement is used to determine in which range the hour belongs and assigns tollAmount with the toll based on the table above, and outputs the toll.

Figure 14.6.1: Calculating toll based on time of day.

```
#include <stdio.h>

int main(void) {
   int timeHour;      // Time of travel hour (24 hour format)
   int timeMinute;    // Time of travel minute
   char inputColon;   // Used to read time format
   double tollAmount;

   printf("Enter time of travel (HH:MM in 24 hour format): ");

   // Read an integer (hour), colon (char), and integer (minute)
   scanf("%d%c%d", &timeHour, &inputColon, &timeMinute);

   // Determine toll based on hour of travel
   if (timeHour < 6) {            // Before 6:00 am
      tollAmount = 1.55;
   }
   else if (timeHour < 10) {    // 6 am to 9:59 am
      tollAmount = 4.65;
   }
   else if (timeHour < 18) {    // 10 am to 5:59 pm
      tollAmount = 2.35;
   }
   else {                         // 6 pm and after
      tollAmount = 1.55;
   }

   // Output time and toll amount
   printf("Toll at %d:", timeHour);

   // Output minute with formatting (discussed elsewhere) to
   // print two digits for minutes.
   printf("%02d is %lf", timeMinute, tollAmount);

   return 0;
}
```

```
Enter time of travel (HH:MM in 24 hour format): 9:30
Toll at 9:30 is 4.650000
```

PARTICIPATION

ACTIVITY 14.6.1: Toll calculation.

For the given input, what is the final value of tollAmount.

1) 5:45

- ○ 0.00
- ○ 1.55
- ○ 2.35

2) 9:45

- ○ 1.55
- ○ 2.35
- ○ 4.65

3) 10:00

- ○ 1.55
- ○ 2.35
- ○ 4.65

4) 22:15

- ○ 1.55
- ○ 2.35

Calculating toll based on time of day and day of week

A toll road oftens has a different toll schedule for weekends and holidays than for weekdays. The table below lists the toll schedule for weekends and holidays.

Table 14.6.2: Toll schedule for weekends and holidays.

Time (am/pm)	Time (24 hour)	Toll amount
Before 8:00 am	Before 8:00	1.55
8:00 am to 11:59 am	8:00 to 11:59	3.05
12:00 pm to 3:59 pm	12:00 to 15:59	3.45
4:00 pm to 6:59 pm	16:00 to 18:59	3.60
7:00 pm to 9:59 pm	19:00 to 21:59	3.05

10:00 pm and after	22:00 and after	1.55

The revised program below additionally gets the type of day from the user (0 for weekdays, and 1 for weekends or holidays). The program uses nested if-else statements to calculate the toll amount. The outer if-else checks if the today is a weekday or weekend/holiday. The nested if-else statements implement the respective toll schedules by determining the appropriate toll based on the hour of travel.

The program also uses if-else statements to output the time of travel using am/pm format instead of 24-hour format.

Figure 14.6.2: Calculating toll based on time of day and day of week.

```
#include <stdio.h>

int main(void) {
   int timeHour;      // Time of travel hour (24 hour format)
   int timeMinute;    // Time of travel minute
   int typeOfDay;     // 0 - weekday, 1 - weekend/holiday
   char inputColon;   // Used to read time format
   double tollAmount;

   printf("Enter time of travel (HH:MM in 24 hour format): ");

   // Read an integer (hour), colon (char), and integer (minute)
   scanf("%d%c%d", &timeHour, &inputColon, &timeMinute);

   printf("Enter type of day (0 - weekday, 1 - weekend/holiday): ");
   scanf("%d", &typeOfDay);

   if (typeOfDay == 0) { // Weekday time and rates
                         // Determine toll based on hour of travel
      if (timeHour < 6) {         // Before 6:00 am
         tollAmount = 1.55;
      }
      else if (timeHour < 10) {   // 6 am to 9:59 am
         tollAmount = 4.65;
      }
      else if (timeHour < 18) {   // 10 am to 5:59 pm
         tollAmount = 2.35;
      }
      else {                      // 6 pm and after
         tollAmount = 1.55;
      }
   }
   else { // Weekend/holiday time and rates
          // Determine toll based on hour of travel
      if (timeHour < 8) {         // Before 8:00 am
         tollAmount = 1.55;
      }
      else if (timeHour < 12) {   // 6 am to 11:59 am
         tollAmount = 3.05;
      }
      else if (timeHour < 16) {   // 12 pm to 3:59 pm
         tollAmount = 3.45;
      }
      else if (timeHour < 19) {   // 4 pm to 6:5 9pm
         tollAmount = 3.60;
      }
      else if (timeHour < 22) {   // 7 pm to 9:59 pm
```

```
         tollAmount = 3.05;
      }
      else {                     // 10 pm and after
         tollAmount = 1.55;
      }
   }

   // Output toll using am/pm format
   printf("Toll at ");

   // Output hour adjusting for am/pm format
   if (timeHour == 0) {
      printf("12:");
   }
   else if (timeHour <= 12) {
      printf("%d:", timeHour);
   }
   else {
      printf("%d:", timeHour - 12);
   }

   // Output minute with formatting (discussed elsewhere) to
   // print two digits for minutes.
   printf("%02d", timeMinute);

   // Output am/pm
   if( timeHour < 12 ) {
      printf(" am");
   }
   else {
      printf(" pm");
   }

   printf(" is %lf\n", tollAmount);

   return 0;
}
```

```
Enter time of travel (HH:MM in 24 hour format): 10:45
Enter type of day (0 - weekday, 1 - weekend/holiday): 1
Toll at 10:45 am is 3.050000
```

PARTICIPATION ACTIVITY 14.6.2: If-else statements for calculating toll amount and formatting time.

Select the value tollAmount is assigned with for the given input.

1) The outer if-else statement checks the type of day, and the nested if-else statements check the hour of travel.

○ True

○ False

2) An alternative implementation that checks the hour of travel in an outer if-else statements and checks the type of day using nested if-else statements

would have the same number of if statements.

- ○ True
- ○ False

3) If timeHour is 0 and timeMinute is 30, the time will be output as: 0:30.

- ○ True
- ○ False

Calculating toll with carpool discount

A toll road may have a discount for carpools, sometimes called high-occupancy vehicles (HOV). The following program uses if-else statement to adjust the toll amount based on the number of person in the vehicle. The carpool discount rules are:

- A carpool is 3 or more person per vehicle.
- The toll for carpools on weekdays between 6:00 am and 10:00 am is half the normal toll.
- Otherwise, the toll for carpools is 0 (as in free).

Figure 14.6.3: Calculating toll with carpool discount.

```
#include <stdio.h>

int main(void) {
   int timeHour;      // Time of travel hour (24 hour format)
   int timeMinute;    // Time of travel minute
   int typeOfDay;     // 0 - weekday, 1 - weekend/holiday
   int numPersons;    // Persons in vehicle
   char inputColon;   // Used to read time format
   double tollAmount;

   printf("Enter time of travel (HH:MM in 24 hour format): ");

   // Read an integer (hour), colon (char), and integer (minute)
   scanf("%d%c%d", &timeHour, &inputColon, &timeMinute);

   printf("Enter type of day (0 - weekday, 1 - weekend/holiday): ");
   scanf("%d", &typeOfDay);

   if (typeOfDay == 0) { // Weekday time and rates
                         // Determine toll based on hour of travel
      if (timeHour < 6) {          // Before 6:00 am
         tollAmount = 1.55;
      }
      else if (timeHour < 10) {    // 6 am to 9:59 am
         tollAmount = 4.65;
      }
      else if (timeHour < 18) {    // 10 am to 5:59 pm
         tollAmount = 2.35;
      }
      else {                       // 6 pm and after
         tollAmount = 1.55;
      }
   }
   else { // Weekend/holiday time and rates
          // Determine toll based on hour of travel
      if (timeHour < 8) {         // Before 8:00 am
         tollAmount = 1.55;
      }
      else if (timeHour < 12) {   // 6 am to 11:59 am
         tollAmount = 3.05;
      }
      else if (timeHour < 16) {   // 12 pm to 3:59 pm
         tollAmount = 3.45;
      }
      else if (timeHour < 19) {   // 4 pm to 6:5 9pm
         tollAmount = 3.60;
      }
      else if (timeHour < 22) {   // 7 pm to 9:59 pm
         tollAmount = 3.05;
      }
      else {                      // 10 pm and after
         tollAmount = 1.55;
      }
   }

   // Check for carpool rate (3 or persons) and update toll
   if (numPersons >= 3) {
      // If on a weekday between 6:00 am and 9:00 am, toll is half off
      if ((typeOfDay == 0) && (timeHour >= 6) && (timeHour < 10)) {
         tollAmount = tollAmount * 0.5;
      }
      // Otherwise, the toll is free
      else {
         tollAmount = 0.0;
      }
   }

   // Output toll using am/pm format
   printf("Toll at ");

   // Output hour adjusting for am/pm format
   if (timeHour == 0) {
      printf("12:");
   }
   else if (timeHour <= 12) {
      printf("%d:", timeHour);
   }
   else {
      printf("%d:", timeHour - 12);
   }

   // Output minute with formatting (discussed elsewhere) to
   // print two digits for minutes.
   printf("%02d", timeMinute);

   // Output am/pm
   if( timeHour < 12 ) {
      printf(" am");
   }
   else {
      printf(" pm");
   }

   printf(" is %lf\n", tollAmount);

   return 0;
}
```

```
Enter time of travel (HH:MM in 24 hour format): 17:15
Enter type of day (0 - weekday, 1 - weekend/holiday): 0
Enter number of persons in vehicle: 3
Toll at 5:15 pm is 0.000000
```

PARTICIPATION ACTIVITY 14.6.3: Toll calculation.

Match the final value of tollAmount to the timeHour, typeOfDay, and numPersons.

4.65 0.0 2.325 1.55

timeHour is 7, typeOfDay is 0, numPersons is 1

timeHour is 8, typeOfDay is 0, numPersons is 4

timeHour is 18, typeOfDay is 1, numPersons is 3

timeHour is 20, typeOfDay is 0, numPersons is 2

Reset

14.7 Loops and strings

Iterating through a string with a for loop

A programmer commonly iterates through a string, examining each character. The following example counts the number of letters in a string, not counting digits, symbols, etc.

Figure 14.7.1: Iterating through a string: Counting letters.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main(void) {
   char inputWord[21];
   int numLetters;
   int i;

   printf("Enter a word (<= 20 char): ");
   scanf("%s", inputWord);

   numLetters = 0;
   for (i = 0; i < strlen(inputWord); ++i) {
      if (isalpha(inputWord[i])) {
         numLetters += 1;
      }
   }

   printf("Number of letters: %d\n", numLetters);

   return 0;
}
```

```
Enter a word (<= 20 char): Hey!!
Number of letters: 3

...

Enter a word (<= 20 char): 123abc...xyz
Number of letters: 6
```

PARTICIPATION ACTIVITY 14.7.1: Iterating through a string.

1) To visit every character in a string, a for loop should iterate from index _____ to index _____ .

- 0 to length
- 0 to length-1
- 1 to length

2) If a for loop iterates through a string s using variable i, the loop body can access the current character as: s[_____]

- i-1
- i
- i+1

Iterating until done with a while loop

The example below appends input words to a string, separating words by spaces, until the word "DONE" is gotten. Because the number of words is unknown beforehand, a while loop is used.

Figure 14.7.2: Iterating until done: Appending words into a sentence.

```
#include <stdio.h>
#include <string.h>

int main(void) {
   char sentenceText[81];
   char currWord[81];
   int numChars;

   printf("Enter words. End with DONE. Max 80
chars.\n");

   strcpy(sentenceText, ""); // Initially empty
   numChars = 0;             // Can't exceed 80

   scanf("%s", currWord);    // Get first word

   while (strcmp(currWord, "DONE") != 0) {
      // Must be <= 79 (plus space is 80) to fit in
string
      if ( (strlen(sentenceText) + strlen(currWord)) >
79) {
         printf("Max sentence length reached.\n");
         strcpy(currWord, "DONE");        // Force done
      }
      else {
         strcat(sentenceText, " ");       // Append a
space
         strcat(sentenceText, currWord);  // Append the
word
         scanf("%s", currWord);           // Get next
word
      }
   }

   printf("Sentence: %s\n", sentenceText);

   return 0;
}
```

```
Enter words. End with DONE. Max 80
chars.
I would like to swim. DONE
Sentence:  I would like to swim.

...

Enter words. End with DONE. Max 80
chars.
Will you
join me?
DONE
Sentence:  Will you join me?
```

PARTICIPATION ACTIVITY 14.7.2: Iterating until done.

1) The number of loop iterations is known before entering the loop.

○ True

○ False

2) Based on user input, the loop possibly might not be entered at all.

○ True

○ False

3) If the call to strcmp returns 0, the loop will be entered.

○ True

○ False

4) In the loop, the user's word is appended to the sentence string, after which the sentence string's length is checked for exceeding 79 characters.

○ True

○ False

5) The user's sentence will begin with a blank space.

○ True

○ False