Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.1

All are valid except the one containing a $ sign.

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.3

```
struct Pair {
  int first;
  double second;
};
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.4

After execution, `s` contains `"abcabcdabc"`. The last seven characters, `"abcdabc"`, arises from operation `s + t[1] + s`, and the first `"abc"` arises from the fact that the assignment uses += to concatenate the contents to `s`.

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.5

(y + (2 * (z ++))) < (3 - (w / 5)).

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.6

Each pointer dp[i] points to a variable that first needs to be allocated before being initialized. Once allocated, we need to use *dp[i] to access the double.

```
double* dp[10]
for (int i = 0; i < 10; i++) {
  dp[i] = new double;
  *dp[i] = 0.0;
}
```

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.7

```
int sumToN(int n) {
  int sum = 0;
  for (int i = 1; i <= n; i++)
    sum += i;
  return sum;
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.8

[bool isMultiple(long n, long m) if (n else return false

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.9

```
void printArray(int** A, int m, int n) {
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      std::cout << A[i][j] << ' ';
    }
    std::cout << endl;
  }
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.10

Both functions produce the same output. Because its argument is called by reference, the function `g` modifies the contents of its actual argument (by incrementing it). In contrast, the argument to function `f` is passed by value, and hence its value does not change.

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.12

```
bool CreditCard::chargeIt(double price) {
  if ((price <= 0) || (price + balance > double(limit)))
    return false;                          //price not positive or limit is met
  balance += price;
  return true;                             // the charge goes through
}

void makePayment(double payment) {
  if (payment <= 0) return;                // ignore negative payment
  balance -= payment;
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.13

This solution assesses a fixed interest rate. A better solution would involve creating an interest rate member variable, which could be adjusted.

```
void makePayment(double payment) {          // pay with interest
  const double interestRate = 0.10;         // 10 percent interest
  if (payment <= 0) return;                 // ignore negative payment
  balance -= payment * (1 + interestRate);
}
```

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.14

Processing of dates would involve a number of additional elements. To simplify things, let us assume that there is a special class Date, which has a comparison function isLaterThan. Each payment transaction is provided with two additional arguments, the due date and the payment date. Finally, we assume a fixed late fee of $10.00.

```
void makePayment(
  double payment,                          // payment amount
  const Date& dueDate,                     // payment due date
  const Date& paymentDate)                 // date of payment
{
  const double lateFee = 10.00;            // 10 dollar late fee
  if (payment <= 0) return;                // ignore negative payment
  balance -= payment;
  if (paymentDate.isLaterThan(dueDate))    // past due?
    balance -= lateFee;
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.15

The following functions can be added to the end of the class definition.

```
class CreditCard {
  // ... add these new modifier functions in the public section
  void setNumber(const string& newNumber) { number = newNumber; }
  void setName(const string& newName) { name = newName; }
  void setBalance(double newBalance) { balance = newBalance; }
  void setLimit(int newLimit) { limit = newLimit; }
};
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.16

```
for (int j=1; j <= 58; j++) {
   wallet[0]->chargeIt(double(i));
   wallet[1]->chargeIt(2.0 * i);
   wallet[2]->chargeIt(double(3 * i));
}
```

This change will cause credit card 2 to go over its limit.

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.17

```
class AllKinds {
private:
  int    intMem;
  long   longMem;
  float  floatMem;
public:
  AllKinds() { intMem = 4; longMem = 23L; floatMem = 3.14159F; }
  void setInt(int i) { intMem = i; }
  void setLong(long l) { longMem = l; }
  void setFloat(float f) { floatMem = f; }
  int getInt() const { return intMem; }
  long getLong() const { return longMem; }
  float getFloat() const { return floatMem; }
  long addIntLong() const { return long(intMem) + longMem; }
  float addIntFloat() const { return float(intMem) + floatMem; }
  float addLongFloat() const { return float(longMem) + floatMem; }
  float addAll() const { return float(intMem) + float(longMem) + floatMem; }
};
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.18

```
bool isMultiple(long n, long m) {
   if (n % m == 0)
      return true;
   else
      return false;
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.19

The following remarkably short and tricky function determines whether a nonnegative integer i is a power of 2.

```
bool isTwoPower(int i) {
  return (i != 0) && (((i−1) & i) == 0);
}
```

The function makes use of the fact that the binary representation of a of $i = 2^k$ is a 1 bit followed by k 0 bits. In this case, the binary representation of $i - 1$ is a 0 bit followed by i 1 bits. Thus, when we take the bitwise "and" of the two bit strings, all the bits cancel out.

$$\begin{aligned} i = 1024_{10} &= 000010000000000_2 \\ i - 1 = 1023_{10} &= 000001111111111_2 \\ i \;\&\; (i-1) &= 000000000000000_2 \end{aligned}$$

If i is not a power of 2 and $i > 0$, then the bit strings for i and $i - 1$ share at least one bit in common, the highest order bit, and so their bitwise "and" will be nonzero. We need to include a special check for zero, since it will pass this test but zero is not a power of 2.

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.20

Here is a solution based on the use of a for loop.

```
int sumToN(int n) {
   int sum = 0;
   for (int i = 1; i < n; i++) sum += i;
   return sum;
}
```

Here is a different solution, based instead on recursion.

```
int sumToN(int n) {
   if (n <= 0)
      return 0;
   else
      return (n−1 + sumToN(n−1));
}
```

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.21

Here is a solution based on the use of a for loop.

```
int sumOdd(int n) {
   int sum = 0;
   for (int i = 1; i < n; i+=2) sum += i;
   return sum;
}
```

Data Structures and Algorithms in C++
(Second Edition)

M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise R-1.22

```
int divideByTwo(double x) {
    int result = 0;
    while (x >= 2) {
        x = x/2;
        result++;
    }
    return result;
}
```

Data Structures and Algorithms in C++
(Second Edition)
M. T. Goodrich, R. Tamassia, and D. M. Mount

Solution of Exercise C-1.3

```
bool allDistinct(const vector<int>& a) {
  for (int i = 0; i < a.size()−1; i++) {
    for (int j = i+1; j < a.size(); j++) {
      if (a[i] == a[j]) return false;
    }
  }
  return true;
}
```