```
        if (c >= 'a' && c <= 'z') c += 'A' - 'a';    // capitalize c
        if (c >= 'A' && c <= 'Z') ++freq[c];             // count c
      }
    }
    cout << "The input had " << lines << " lines, " << words
         << " words,\nand the following letter frequencies:\n";
    for (int i=65; i<SIZE; i++)
    { cout << '\t' << char(i) << ": " << freq[i];
      if (i > 0 && i%8 == 0) cout << endl;      // print 8 to a line
    }
    cout << endl;
}
```

9.7

```
bool is_upper(char c)
{ return bool(c >= 'A' && c <= 'Z');
}
bool is_lower(char c)
{ return bool(c >= 'a' && c <= 'z');
}
bool is_letter(char c)
{ return bool(is_upper(c) || is_lower(c));
}
void reduce(string& s)
{ while (s.length() > 0 && !is_letter(s[0]))
    s.erase(0, 1);
  int k = s.length() - 1;
  while (k > 0 && !is_letter(s[k--]))
    s.erase(k+1, 1);
  int len = s.length();
  if (len == 0) return;
  for (int i=0; i<len; i++)
    if (is_upper(s[i])) s[i] += 'a' - 'A';
}
```

9.8

```
int main()
{ ifstream in("Pr0907.in");
  string s;
  const int SIZE=1000;     // assume at most 1000 different words
  string word[SIZE];                          // holds words read
  int lines=0, words=0, n=0, freq[SIZE]={0}, i;
  char c;
  while (in >> s)
  { reduce(s);
    if (s.length() == 0) continue;
    ++words;
    in.get(c);
    if (c == '\n') ++lines;                        // count line
    for (i=0; i<n; i++)
      if (word[i] == s) break;
    if (i == n) word[n++] = s;                // add word to list
    ++freq[i];                                      // count word
  }
  cout << "The input had " << lines << " lines and " << words
       << " words,\nwith the following frequencies:\n";
```

```
  for (int i=0; i<n; i++)
  { s = word[i];
    if (i > 0 && i%3 == 0) cout << endl;      // print 3 to a line
    cout << setw(16) << setiosflags(ios::right)
         << s.c_str() << ": " << setw(2) << freq[i];
  }
  cout << endl;
}
```

9.9

```
int main()
{ const int SIZE=100;  // maximum number of lines stored
  string line[SIZE], s;
  int n=0, len, maxlen=0;
  while (!cin.eof())
  { getline(cin, s);
    len = s.length();
    if (len > 0) cout << s << endl;
    if (len > maxlen) maxlen = len;
    line[n++] = s;
  }
  --n;                      // n == number of lines read
  for (int i=0; i<n; i++)
  { s = line[i];
    len = s.length();
    cout << string(maxlen-len, ' ') << s << endl;
  }
}
```

9.10

```
string Roman(int n)
{ int d3 = n/1000;  // the thousands digit
  string s(d3, 'M');
  n %= 1000;
  int d2 = n/100;  // the hundreds digit
  if (d2 == 9) s += "CM";
  else if (d2 >= 5)
  { s += "D";
    s += string(d2-5, 'C');
  }
  else if (d2 == 4) s += "CD";
  else s += string(d2, 'C');
  n %= 100;
  int d1 = n/10;  // the tens digit
  if (d1 == 9) s += "XC";
  else if (d1 >= 5)
  { s += "L";
    s += string(d1-5, 'X');
  }
  else if (d1 == 4) s += "XL";
  else s += string(d1, 'X');
  n %= 10;
  int d0 = n/1;  // the ones digit
  if (d0 == 9) s += "IX";
  else if (d0 >= 5)
  { s += "V";
```

```
      s += string(d0-5, 'I');
    }
    else if (d0 == 4) s += "IV";
    else s += string(d0, 'I');
    return s;
  }
```

9.11

```
  int v(string s, int i)
  { char c = s[i];
    if (c == 'M') return 1000;
    if (c == 'D') return 500;
    if (c == 'C') return 100;
    if (c == 'L') return 50;
    if (c == 'X') return 10;
    if (c == 'V') return 5;
    if (c == 'I') return 1;
    return 0;
  }
  int HindArabic(string s)
  { int n0=0, n1=0, n=0;
    for (int i=0; i<s.length(); i++)
    { n0 = n1;
      n += n1 = v(s,i);
      if (n1>n0) n -= 2*n0;
    }
    return n;
  }
```

9.12

```
  char c(int k)
  { assert(k >= 0 && k <= 15);
    if (k < 10) return char(k + '0');
    return char(k - 10 + 'a');
  }
  string hexadecimal(int n)
  { if (n == 0) return string(1, '0');
    string s;
    while (n > 0)
    { s = string(1, c(n%16)) + s;
      n /= 16;
    }
    return s;
  }
```

9.13

```
  int v(string s, int i)
  { char c = s[i];
    assert(c >= '0' && c <= '9' || c >= 'a' && c <= 'f');
    if (c >= '0' && c <= '9') return int(c - '0');
    else return int(c - 'a' + 10);
  }
  int decimal(string s)
  { int len = s.length();
    assert(len > 0);
    int n=0;
    for (int i=0; i<len; i++)
      n = 16*n + v(s,i);
```

```
  return n;
}
```

9.14

```
void reverse(string& s)
{ string temp = s;
  int len = s.length();
  for (int i=0; i<len; i++)
    s[i] = temp[len-i-1];
}
```

9.15

```
bool is_palindrome(string s)
{ int len = s.length();
  for (int i=0; i<len/2; i++)
    if (s[i] != s[len-i-1]) return false;
  return true;
}
```

9.16

```
bool more(ifstream& fin, string& s)
{ if (getline(fin, s)) return true;
  else return false;
}

bool copy(ofstream& fout, ifstream& fin, string& s)
{ fout << s << endl;
  cout << s << endl;
  return more(fin,s);
}
int main()
{ ifstream fin1("Democrats.dat");
  ifstream fin2("Republicans.dat");
  ofstream fout("Presidents.dat");
  string s1, s2;
  bool more1 = more(fin1, s1);
  bool more2 = more(fin2, s2);
  while (more1 && more2)
    if (s1 < s2) more1 = copy(fout, fin1, s1);
    else more2 = copy(fout, fin2, s2);
  while (more1)
    more1 = copy(fout, fin1, s1);
  while (more2)
    more2 = copy(fout, fin2, s2);
  fout << endl;
}
```