successor. If the two values are not in the proper order relative to each other, they are swapped. The array will be completely sorted when no swaps are made during a pass. Write a C++ version of this algorithm. Then subject your algorithm to a big-O efficiency analysis as we did for the selection and insertion sorts. Compare bubble sort's best, average, and worst case performance to those of the other two algorithms.

5. Do a big-O analysis for those statements inside each of the following nested loop constructs.

a.
```
for (k = 0; k < n; ++k)
  for (j = 6; j < n; ++j)
  {
    ...
  }
```

b.
```
for (k = 0; k < n; ++k)
{
  j = n;
  while (j > 0)
  {
    ...
    j /= 2;       // integer division
  }
}
```

c.
```
k = 1;
do
{
  j = 1;
  do
  {
    ...
    j *= 2;
  }
  while (j < n);
  ++k;
}
while (k < n);
```

6. An algorithm has an efficiency $O(n^2|\sin(n)|)$. Is it any better than $O(n^2)$ for large integer n? Explain why, or why not.

7. Suppose that each of the following expressions represents the number of logical operations in an algorithm as a function of n, the number of data items being manipulated. For each expression, determine the dominant term and then classify the algorithm in big-O terms.

   a. $n^3 + n^2\log_2 n + n^3\log_2 n$
   b. $n + 4n^2 + 4^n$
   c. $48n^4 + 16n^2 + \log_8 n + 2n$

8. Consider the following nested loop construct. Categorize its efficiency in terms of the variable ***n*** using big-O notation. Suppose the statements represented by the ellipsis require four main memory accesses (each requiring one microsecond) and two disk file accesses (each requiring one millisecond). Express in milliseconds the amount of time this construct would require to execute if ***n*** were 1000.

```
x = 1;
do
{
  y = n;
  while (y > 0)
  {
    ...
    --y;
  }
  x *= 2;
}
while (x < n*n);
```

9. A sorting algorithm is called *stable* if it does not change the relative order of array elements that are equal. For example, a stable sorting algorithm will not place 13_1 after 13_2 in the array

18	13	6	12	13	9

Are insertion sort and selection sort stable? If not, provide an example of an array with at least two equal elements that change in their order relative to each other.

1.3 Algorithm Efficiency—The Search Problem

In the previous section, we introduced big-O notation and used it to analyze two algorithmic solutions to the sorting problem. In this section we examine a general strategy for solving the search problem—finding a particular value in an ordered array. As in Section 1.2, we may state this problem more formally as a C++ interface expressed in precondition/postcondition form.

successor. If the two values are not in the proper order relative to each other, they are swapped. The array will be completely sorted when no swaps are made during a pass. Write a C++ version of this algorithm. Then subject your algorithm to a big-O efficiency analysis as we did for the selection and insertion sorts. Compare bubble sort's best, average, and worst case performance to those of the other two algorithms.

5. Do a big-O analysis for those statements inside each of the following nested loop constructs.

a.
```
for (k = 0; k < n; ++k)
  for (j = 6; j < n; ++j)
  {
    ...
  }
```

b.
```
for (k = 0; k < n; ++k)
{
  j = n;
  while (j > 0)
  {
    ...
    j /= 2;        // integer division
  }
}
```

c.
```
k = 1;
do
{
  j = 1;
  do
  {
    ...
    j *= 2;
  }
  while (j < n);
  ++k;
}
while (k < n);
```

6. An algorithm has an efficiency $O(n^2|\sin(n)|)$. Is it any better than $O(n^2)$ for large integer n? Explain why, or why not.

7. Suppose that each of the following expressions represents the number of logical operations in an algorithm as a function of n, the number of data items being manipulated. For each expression, determine the dominant term and then classify the algorithm in big-O terms.

   a. $n^3 + n^2 \log_2 n + n^3 \log_2 n$

   b. $n + 4n^2 + 4^n$

   c. $48n^4 + 16n^2 + \log_8 n + 2n$

8. Consider the following nested loop construct. Categorize its efficiency in terms of the variable n using big-O notation. Suppose the statements represented by the ellipsis require four main memory accesses (each requiring one microsecond) and two disk file accesses (each requiring one millisecond). Express in milliseconds the amount of time this construct would require to execute if n were 1000.

```
x = 1;
do
{
  y = n;
  while (y > 0)
  {
    ...
    --y;
  }
  x *= 2;
}
while (x < n*n);
```

9. A sorting algorithm is called *stable* if it does not change the relative order of array elements that are equal. For example, a stable sorting algorithm will not place 13_1 after 13_2 in the array

18	13	6	12	13	9

Are insertion sort and selection sort stable? If not, provide an example of an array with at least two equal elements that change in their order relative to each other.

1.3 Algorithm Efficiency—The Search Problem

In the previous section, we introduced big-O notation and used it to analyze two algorithmic solutions to the sorting problem. In this section we examine a general strategy for solving the search problem—finding a particular value in an ordered array. As in Section 1.2, we may state this problem more formally as a C++ interface expressed in precondition/postcondition form.