```
-----------------------------------------------------------------
sort_algorithms_1.h: selection, bubble, and insertion sort
-----------------------------------------------------------------

#ifndef __SORT_1_H__
#define __SORT_1_H__

#include <vector>

template <typename T>
void selection_sort(std::vector<T> &A) {
  int i, j, k, N=A.size();

  for (i=0; i<N-1; i++) {
    k = i;
    for (j=i+1; j<N; j++) {
      if (A[j] < A[k])
        k = j;
    }
    swap(A[i], A[k]);
  }
}

template <typename T>
void bubble_sort(std::vector<T> &A) {
  int i, j, N=A.size();

  for (i=N-1; 0<i; i--) {
    for (j=0; j<i; j++) {
      if (A[j+1] < A[j])
        swap(A[j], A[j+1]);
    }
  }
}

template <typename T>
void insertion_sort(std::vector<T> &A) {
  int i, j, N=A.size();

  for (i=1; i<N; i++) {
    T tmp = A[i];
    for (j=i; 0<j && tmp<A[j-1]; j--)
      A[j] = A[j-1];
    A[j] = tmp;
  }
}

#endif
```

```
-----------------------------------------------------------------
sort_usage.cpp: driver code for testing sort algorithms
-----------------------------------------------------------------

#include <...>
using namespace std;

#include "sort_algorithms_1.h"

template <typename T>
void readdata(string &fname, vector<T> &A) {
  ifstream fin(fname.c_str());

  T din;
  while (fin >> din)
    A.push_back(din);

  fin.close();
}

template <typename T>
void sortdata(vector<T> &A, string &algname) {
  if (algname.compare("bubble") == 0)
    bubble_sort(A);

  else if (algname.compare("selection") == 0)
    selection_sort(A);

  else if (algname.compare("insertion") == 0)
    insertion_sort(A);
}

template <typename T>
void printdata(T p1, T p2, string &fname) {
  ofstream fout(fname.c_str(), fstream::trunc);

  while (p1 != p2) {
    fout << *p1 << "\n";
    ++p1;
  }

  fout.close();
}

-----------------------------------------------------------------
Hint: Functions sortdata() and main() will be updated in future
classes to include sorting algorithms covered then. Other driver
functions remain as shown here.
-----------------------------------------------------------------
```

```
int main(int argc, char *argv[]) {
  if (argc != 3) {
    cerr << "usage: " << argv[0]
         << " -selection|bubble|insertion"
         << " file.txt\n";
    return 0;
  }

  string algname(&argv[1][1]);
  string fname_in(argv[2]);

  vector<string> A;

  readdata(fname_in, A);
  sortdata(A, algname);

  string fname_out = algname + "_" + fname_in;

  printdata(A.begin(), A.end(), fname_out);
}
```

---

find_usage.cpp: driver code for testing sorting + binary search

---

```
#include <...>
using namespace std;

#include "sort_algorithms_1.h"

template <typename T>
int find(vector<T> &A, T &target) {
  int left=0, right=A.size()-1, middle;

  while (left <= right) {
    middle = (left + right)/2;

    if (target < A[middle])
      right = middle - 1;
    else
    if (A[middle] < target)
      left = middle + 1;
    else
      return middle;
  }

  return -1;
}

template <typename T>
void readdata(string &fname, vector<T> &A) { ... }

template <typename T>
void sortdata(vector<T> &A) { insertion_sort(A); }

int main(int argc, char *argv[]) {
  if (argc != 2) {
    cerr << "usage: " << argv[0] << " file.txt\n";
    return 0;
  }

  string fname_in(argv[1]);

  vector<string> A;
  string text;

  readdata(fname_in, A);
  sortdata(A);

  while (1) {
    cout << "Find> ";

    cin >> text;
    if (cin.eof())    break;

    int index = find(A, text);
    if (index != -1)
      cout << index+1 << ":" << text << "\n\n";
  }

  cout << "\n";
}
```

```
unix> ./sort_usage -insertion cities.txt
unix> cat -n insertion_cities.txt
     1  Chattanooga
     2  Knoxville
     3  Memphis
     4  Nashville
     5  Wartburg
unix> ./find_usage insertion_cities.txt
Find> Oak_Ridge
Find> Knoxville
2
Find> Nashville
4
```