Problem: Develop **class** *List* that reads, stores, sorts, and prints a list of strings. Specifically, we want to do the following:

1. Read a list of strings from file **i01.dat**
2. Print the list in the order that the strings were read.
3. Sort the list.
4. Print the list after it has been sorted

For example, suppose file **i01.dat** contains the strings shown in figure 1.

```
Elise Ilse Belle Gabrielle Diana Andrea Fantine Hester Cosette
```

Figure 1. Strings in file **i01.dat**.

We want our program to print the same strings in alphabetical order as shown in Figure 2.

```
{Andrea,Belle,Cosette,Diana,Elise,Fantine,Gabrielle,Hester,Ilse}
```

Figure 2. Alphabetized list.

The technique for developing and debugging a program is to start with a program that works. Then, carefully and methodically test *small* increments to the program. This process is demonstrated below.

Start with the simplest program that works as shown in figure 3. Compile file **p01.cpp.** Execute program **p01**.

```
int main()
{
    return 0;
}
```

Figure 3. File **p01.cpp** version 1.

Add a declaration and supporting include files that will permit the program to read file **i01.dat**. Compile file **p01.cpp.** Execute program **p01**.

```
#include <fstream>
using namespace std;
int main()
{   ifstream i;
    return 0;
}
```

Figure 4. File **p01.cpp** version 2.

Declare the name of the file and open the file. Compile file **p01.cpp.** Execute program **p01**.

```
#include <fstream>
using namespace std;
int main()
{    char ifn[]="i01.dat";
     ifstream i(ifn);
     return 0;
}
```

Figure 5. File **p01.cpp** version 3.

Declare the name of the file and open the file. Compile file **p01.cpp.** Execute program **p01**.

```
#include <fstream>
using namespace std;
int main()
{    char ifn[]="i01.dat";
     ifstream i(ifn);
     return 0;
}
```

Figure 5. File **p01.cpp** version 3.

Put in a test that determines if file **i01.dat** exists. Add C++ include file *iostream* for the standard output, *cout*. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{    char ifn[]="i01.dat";
     ifstream i(ifn);
     if (!i) cout << "File " << ifn << " could not be opened.";
     return 0;
}
```

Figure 6. File **p01.cpp** version 4.

Program **p01** produces:

```
File i01.dat could not be opened.
```

The reason file **i01.dat** could not be opened is because file **i01.dat** does not exist. Create file **i01.dat**. Store the strings shown in figure 1 in file **i01.dat**. Execute program **p01** again and observe that the program does not print the message.

Create function *ListMgr* and invoke the function in function *main*. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
using namespace std;
void ListMgr(istream& i,ostream& o)
{
}
int main()
{    char ifn[]="i01.dat";
     ifstream i(ifn);
     if (!i) cout << "File " << ifn << " could not be opened.";
     ListMgr(i,cout);
     return 0;
}
```

Figure 7. File **p01.cpp** version 5.

Create **class** *List* and invoke the default constructor for **class** *List*. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
using namespace std;
class List {
};
void ListMgr(istream& i,ostream& o)
{    List L;
}
int main()
{    char ifn[]="i01.dat";
     ifstream i(ifn);
     if (!i) cout << "File " << ifn << " could not be opened.";
     ListMgr(i,cout);
     return 0;
}
```

Figure 8. File **p01.cpp** version 6.

Declare the data members of **class** *List*. Do not forget to include standard C++ include file *string*.. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class List {
    int size;       //Number of available elements
    int count;      //Number of occupied elements
                    //Index of the next available element
    string* L;      //Points to a dynamically allocated array of strings
};
void ListMgr(istream& i,ostream& o)
{   List L;
}
int main()
{   char ifn[]="i01.dat";
    ifstream i(ifn);
    if (!i) cout << "File " << ifn << " could not be opened.";
    ListMgr(i,cout);
    return 0;
}
```

Figure 9. File **p01.cpp** version 7.

Declare the data members of **class** *List*. Do not forget to include standard C++ include file *string*.. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class List {
    int size;       //Number of available elements
    int count;      //Number of occupied elements
                    //Index of the next available element
    string* L;      //Points to a dynamically allocated array of strings
public:
    List():size(100),count(0){L=new string[size];}
};
void ListMgr(istream& i,ostream& o)
{   List L;
}
int main()
{   char ifn[]="i01.dat";
    ifstream i(ifn);
    if (!i) cout << "File " << ifn << " could not be opened.";
    ListMgr(i,cout);
    return 0;
}
```

Figure 10. File **p01.cpp** version 8.

Write and test member function *Print*. The list is empty so only an opening and closing brace are produced. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class List {
    int size;       //Number of available elements
    int count;      //Number of occupied elements
                    //Index of the next available element
    string* L;      //Points to a dynamically allocated array of strings
public:
    List():size(100),count(0){L=new string[size];}
    void Print(ostream& o)
    {   o << "{";
        for (int a=0;a<count;a++) {
            if (a>0) o << ",";
            o << L[a];
        }
        o << "}";
    }
};
void ListMgr(istream& i,ostream& o)
{   List L;
    o << endl;
    L.Print(o);
    o << endl;
}
int main()
{   char ifn[]="i01.dat";
    ifstream i(ifn);
    if (!i) cout << "File " << ifn << " could not be opened.";
    ListMgr(i,cout);
    return 0;
}
```

Figure 11. File **p01.cpp** version 9.

Program **p01** produces:

```
{}
```

Write and test member function *Dump*. Function *Dump* prints the value of private members. Note that the call to function *Print* has been replaced by a call to function *Dump*. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class List {
    int size;       //Number of available elements
    int count;      //Number of occupied elements
                    //Index of the next available element
    string* L;      //Points to a dynamically allocated array of strings
public:
    List():size(100),count(0){L=new string[size];}
    void Print(ostream& o)
    {   o << "{";
        for (int a=0;a<count;a++) {
            if (a>0) o << ",";
            o << L[a];
        }
        o << "}";
    }
    void Dump(ostream& o)
    {   o << "size=" << size << " count=" << count << " ";
        Print(o);
    }
};
void ListMgr(istream& i,ostream& o)
{   List L;
    o << endl;
    L.Dump(o);
    o << endl;
}
int main()
{   char ifn[]="i01.dat";
    ifstream i(ifn);
    if (!i) cout << "File " << ifn << " could not be opened.";
    ListMgr(i,cout);
    return 0;
}
```

Figure 12. File **p01.cpp** version 10.

Program **p01** produces:

```
size=100 count=0 {}
```

Write and test member function *Insert*. Function Insert must ensure that space is available for the string to be inserted. Function *Insert* requires the existence of function *IsFull* and constructor *ListFullException*. Write function *IsFull* and the empty **class** *ListFullException*. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class List {
    int size;       //Number of available elements
    int count;      //Number of occupied elements
                    //Index of the next available element
    string* L;      //Points to a dynamically allocated array of strings
public:
    class ListFullException{};
    List():size(100),count(0){L=new string[size];}
    void Print(ostream& o)
    {   o << "{";
        for (int a=0;a<count;a++) {
            if (a>0) o << ",";
            o << L[a];
        }
        o << "}";
    }
    void Dump(ostream& o)
    {   o << "size=" << size << " count=" << count << " ";
        Print(o);
    }
    bool IsFull(void){return count>=size;}
    void Insert(string v)
    {   if (IsFull()) throw ListFullException();
        L[count++]=v;
    }
};
void ListMgr(istream& i,ostream& o)
{   List L;
    L.Insert("Abigail");
    o << endl;
    L.Dump(o);
    o << endl;
}
int main()
{   char ifn[]="i01.dat";
    ifstream i(ifn);
    if (!i) cout << "File " << ifn << " could not be opened.";
    ListMgr(i,cout);
    return 0;
}
```

Figure 13. File **p01.cpp** version 11.

Program **p01** produces:

```
size=100 count=1 {Abigail}
```

Write and test member function *Scan*. Function *Scan* reads and inserts strings read from stream i into the list. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class List {
      int size;          //Number of available elements
      int count; //Number of occupied elements
                         //Index of the next available element
      string* L; //Points to a dynamically allocated array of strings
public:
      class ListFullException{};
      List():size(100),count(0){L=new string[size];}
      void Print(ostream& o)
      {     o << "{";
            for (int a=0;a<count;a++) {
                  if (a>0) o << ",";
                  o << L[a];
            }
            o << "}";
      }
      void Dump(ostream& o)
      {     o << "size=" << size << " count=" << count << " ";
            Print(o);
      }
      bool IsFull(void){return count>=size;}
      void Insert(string v)
      {     if (IsFull()) throw ListFullException();
            L[count++]=v;
      }
      void Scan(istream& i)
      {     for (;;) {
                  string v;
                  i >> v;
                  if (i.eof()) break;
                  Insert(v);
            }
      }
};
void ListMgr(istream& i,ostream& o)
{     List L;
      L.Scan(i);
      o << endl;
      L.Dump(o);
      o << endl;
}
int main()
{     char ifn[]="i01.dat";
      ifstream i(ifn);
      if (!i) cout << "File " << ifn << " could not be opened.";
      ListMgr(i,cout);
      return 0;
}
```

Figure 14. File **p01.cpp** version 12.

Program **p01** produces:

```
size=100 count=9 {Elise,Ilse,Belle,Gabrielle,Diana,Andrea,Fantine,Hester,Cosette}
```

Write and test member functions *Swap* and *Sort*. Functions *Sort* and *Swap* sort the list. Note that the call to function *Dump* has been replaced by a call to function *Print*. Compile file **p01.cpp.** Execute program **p01**.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class List {
      int size;          //Number of available elements
      int count; //Number of occupied elements
                         //Index of the next available element
      string* L; //Points to a dynamically allocated array of strings
public:
      class ListFullException{};
      List():size(100),count(0){L=new string[size];}
      void Print(ostream& o)
      {     o << "{";
            for (int a=0;a<count;a++) {
                  if (a>0) o << ",";
                  o << L[a];
            }
            o << "}";
      }
      void Dump(ostream& o)
      {     o << "size=" << size << " count=" << count << " ";
            Print(o);
      }
      bool IsFull(void){return count>=size;}
      void Insert(string v)
      {     if (IsFull()) throw ListFullException();
            L[count++]=v;
      }
      void Scan(istream& i)
      {     for (;;) {
                  string v;
                  i >> v;
                  if (i.eof()) break;
                  Insert(v);
            }
      }
      void Swap(string& m,string& w){string b=m;m=w;w=b;}
      void Sort(void)
      {     for (int eol=count-1;eol>0;eol--) {
                  int iom=0;
                  for (int i=1;i<=eol;i++) if (L[i]>L[iom]) iom=i;
                  Swap(L[iom],L[eol]);
            }
      }
};
void ListMgr(istream& i,ostream& o)
{     List L;
      L.Scan(i);
      L.Sort();
      o << endl;L.Print(o);o << endl;
}
int main()
{     char ifn[]="i01.dat";
      ifstream i(ifn);
      if (!i) cout << "File " << ifn << " could not be opened.";
      ListMgr(i,cout);
      return 0;
}
```

Figure 15. File **p01.cpp** version 13.

Program **p01** produces:

```
{Andrea,Belle,Belle,Andrea,Diana,Andrea,Fantine,Hester,Cosette}
```