```
---------------------------------------------------------------------
array_list1.cpp: list code using explicit array implementation
---------------------------------------------------------------------

#include <algorithm>
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
  int Nmax=8;
  int *list = new int[Nmax];

  int din;
  int N=0;
  while (cin >> din) {
    if (N == Nmax) {
      int M = 2*Nmax;
      int *n_list = new int[M];

      for (int i=0; i<N; i++)
        n_list[i] = list[i];

      delete [] list;
      list = n_list;

      Nmax = M;
    }

    list[N++] = din;
  }

  sort(&list[0], &list[N]);

  for (int i=0; i<N; i++)
    cout << list[i] << " ";
  cout << "\n";

  delete [] list;
}
```

```
---------------------------------------------------------------------
array_list2.cpp: list code using class based array implementation
---------------------------------------------------------------------

#include <...>
using namespace std;

template <typename T>
class array_list {
  public:
    array_list() { Nmax=8; N=0; data = new T[Nmax]; }
    ~array_list() { delete [] data; }

    int size() { return N; }
    T & operator[](int i) { return data[i]; }

    void push_back(const T &);

  private:
    int Nmax;
    int N;
    T *data;

  public:
    typedef T * iterator;
    iterator begin() { return &data[0]; }
    iterator end() { return &data[N]; }
};

template <typename T>
void array_list<T>::push_back(const T &din) {
  if (N == Nmax) { resize -- see array_list1.cpp }

  data[N++] = din;
}

int main(int argc, char *argv[]) {
  array_list<int> list;

  int din;
  while (cin >> din)
    list.push_back(din);

  sort(list.begin(), list.end());

  array_list<int>::iterator p;
  for (p=list.begin(); p != list.end(); ++p)
    cout << *p << "\n";
}
```

```
-----------------------------------------------------------------
array_list3.cpp: list code using STL array implementation
-----------------------------------------------------------------

#include <...>
using namespace std;

int main(int argc, char *argv[]) {
  vector<int> list;

  int din;
  while (cin >> din)
    list.push_back(din);

  sort(list.begin(), list.end());

  vector<int>::iterator p;
  for (p=list.begin(); p != list.end(); ++p)
    cout << *p << "\n";
}


-----------------------------------------------------------------
Hint: Algorithms like std::sort apply to any list that provides
random access, be that thru pointers or iterators.

Hint: User defined data can be processed provided the less-than
comparison operator has been overloaded (defined). You will do
that in many labs.

Hint: Only functionality needed to support the application shown
here is implemented by the array_list class. A true list class
like std::vector provide many more capabilities.

Hint: The iterator subclass can be replaced by a pointer typedef
since iterators would merely represent array pointers anyway.

-----------------------------------------------------------------

Heads-up: For Lab 1, you will use an std::vector list similar to
array_list3.cpp, but you will also write explicit linked list
code along the lines of array_list1.cpp where the code is written
where needed (as opposed to be being packed up in a class).

-----------------------------------------------------------------
```