```cpp
#include <iostream>
#include <string>

#include "data.h"

const data_t& data_t::operator=(const data_t& rhs)
{
  if(this != &rhs) {
    name = rhs.name;
    id = rhs.id;
  }
  return *this;
}

std::ostream& operator<<(std::ostream& s, const data_t& rhs)
{
  s << rhs.name << " " << rhs.id << std::endl;

  return s;
}
```

```
#include <iostream>
#include <string>
#include <list>

#include "data.h"

int main(int argc, char *argv[]);

int main(int argc, char *argv[])
{
        std::list<data_t* >                     list;
        std::list<data_t* >::iterator           itr;
        std::list<data_t* >::reverse_iterator   ritr;
        data_t                                  *data;
        std::string                             name;
        int                                     num;

  // read input file consisting of names and id codes adding entities to list
  while(!std::cin.eof()) {
    std::cin >> name >> num;
    if(std::cin.good()) {
    //std::cout << name << " " << num << std::endl;
      data = new data_t(name,num);
      std::cout << data;
      list.push_back(data);
    }
  }

  // now play it back
  std::cout << "*** printing list" << std::endl;

  for(itr = list.begin(); itr != list.end(); ++itr) {
    data = (data_t *)*itr;
    std::cout << data;
  }

  // try it again...
  std::cout << "*** printing list again" << std::endl;

  for(itr = list.begin(); itr != list.end(); ++itr) {
    data = (data_t *)*itr;
    std::cout << data;
  }

  // now free all list control structures and the data_t structures as well
  std::cout << "*** deleting list" << std::endl;
  list.clear();

  // see if we trashed malloc's control structures
  std::cout << "*** initializing new list" << std::endl;

  // nope ... see if we can delete an empty list
  std::cout << "*** deleting list again" << std::endl;
  list.clear();

  // prove we survived
  std::cout << "*** done" << std::endl;
}
```