```
    typedef double XFLOAT;
    typedef double OTA_FLOAT;

using namespace std;

namespace SQFUNCS_POLQA_INTERNAL
{

SQStorage::SQStorage()
{
}

SQStorageSkeletonClass* SQStorage::Get(string const &id)
{
    for (int i = 0; i < (int)lItems.size(); i++)
        if (lItems[i].name == id)
            return lItems[i].ptrData;
    return NULL;
}

void SQStorage::Store(string const &id, SQStorageSkeletonClass* data)
{
    if (HasData(id))
        OPTTHROW(( string("ERROR in SQStorage::Store: Data with this ID already present
in the SQStorage singleton.\n")));
    lItems.push_back(SQStorageItem(id, data));
}

void SQStorage::Clear(string const &id)
{
    bool success = false;
    vector<struct SQStorageItem>::iterator curItem = lItems.begin();
    while(curItem != lItems.end())
    {
        if (curItem->name == id)
        {
            delete curItem->ptrData;
            curItem->ptrData = NULL;
            lItems.erase(curItem);
            success = true;
            break;
        }
        curItem++;
    }
    if (!success)
        OPTTHROW(( string("ERROR in SQStorage::Clear: Item \"" + id + "\" could not be
found in the storage.\n")));

}

void SQStorage::ClearAllItemsWhoseIDStartsWith(string const &id)
{
    int j;
    std::vector<struct SQStorageItem>::iterator it;

    for (int i = 0; i < (int)lItems.size(); i++)
    {
        if (lItems[i].name.size() >= id.size() &&
            lItems[i].name.compare(0, id.size(), id) == 0)
        {
            delete lItems[i].ptrData;
            lItems[i].ptrData = NULL;
            for (it = lItems.begin(), j = 0;
                 it != lItems.end() && j != i;
                 j++, it++);
            if (it == lItems.end())
                    OPTTHROW ((string("Internal error in
SQStorage::ClearAllItemsWhoseIDStartsWith!\n")));

            lItems.erase(it);
            i--;
        }
    }
```

```
}

void SQStorage::ClearAll()
{
    for (int i = 0; i < (int)lItems.size(); i++)
    {
        delete lItems[i].ptrData;
        lItems[i].ptrData = NULL;
    }
    lItems.clear();
}

bool SQStorage::HasData(string const &id)
{
    for (int i = 0; i < (int)lItems.size(); i++)
        if (lItems[i].name == id)
            return true;
    return false;
}

}
```