EC327 - "Introduction to Software Engineering"
Lab 10 – Inheritance, Templates, Exceptions
Mo, 04/13/15 - Fr, 04/17/15

In Lab 10, we define two problems (P1, P2). To get full credits from this lab, **you are required to solve both problems**. Solving only P1 will result in partial credits. You should demonstrate and explain your solution approach for the problems. Ensure that your code compiles and executes properly before your demonstration.

GOOD LUCK!

P1 – User & Admin

A generic user class is provided in *User.h* and *User.cpp* files. Admin is a type of User, where an Admin could potentially be a root user (who has all privileges). Derive an Admin class based on the User class using inheritance and add the following fields and functionality:

1. Add a boolean data field `isRoot`.
2. Implement a get function to read the `isRoot` field.
3. Implement a **single** constructor that inputs the name (string), disk quota (int), and an **optional** field isRoot (bool), which is false by default.
4. `addData()` function should have the same functionality as in the User class unless the `isRoot` variable is true. If `isRoot` is true, `addData()` should add the data regardless of the available quota (i.e., root can override the quota, and the available quota can become negative).
5. `getInfo` function should return the string "Admin".

In the *User.h* file, you can modify the functions to be virtual functions and modify the accessibility of the data fields as needed. Use *P1.cpp* to test your Admin class. You may not modify *P1.cpp* file.

P2 – Extending Quota

In this problem, you will use the User and Admin classes you have used in P1 **without any modifications**. Using the function below, create a **vector** of 100 pointers to randomized User objects (you will need to add -std=c++0x as an argument to g++).

```
#include <cstdlib>
//returns a pointer to a randomized User object
User* randUser()
{
    User* u;
    static int ID = 0;
    int quota = 5000 + rand() % 10000;
    string name = "name" + to_string((long long)ID++);
    if((rand() % 100) % 10 == 0){
        if(rand() % 2 == 0){
            u = new Admin(name, quota, true);
        } else {
            u = new Admin(name, quota);
        }
    } else {
        u = new User(name, quota);
    }
    u->addData(quota * rand()%256 / 256);
    return u;
}
```

After creating the vector, you are required to calculate an extended quota for each user as follows:

- The root users don't get any quota extension.
- The quota of non-root admins will be multiplied by 4.
- The quota of regular users will be multiplied by 2.

Print the new quota of each user in the vector with the function below. You do not need to update the diskQuota variable in the User objects.

```
void printUpdatedQuota(const User & u, int extendedQuota)
{
    cout << u.getInfo() << "\t"
         << u.getDiskQuota() << "\t"
         << updatedQuota << endl;
}
```

Hint: You need to use dynamic or static casting to figure out whether a User is an Admin.