class product

{

protected:

…

public:

void addProduct(product p1)

void deleteProduct(int ID)

product searchProduct(int ID)

}

struct person

{

protected:

…

}

class manager

{

this class will inherit all of its attributes, name, id, gender, etc, from the person struct

public:

void addPerson(person p1)

void deletePerson(int ID)

product searchPerson(int ID)

void addProductToInventory()

}

int main

{

Some form of authentication

if (person is manager)

{

able to call following functions:

void addProduct(product p1) - The program will ask the manager for title, price, etc. This info will then be used to create a new object of type product which will then be passed as a parameter to the addProduct function, all of this can be done by the function at the bottom of this document.

void deleteProduct(int ID)

product searchProduct(int ID)

void addPerson(person p1)

void deletePerson(int ID)

product searchPerson(int ID)

}

if (person is employee)

{

able to call following functions:

void addProduct(product p1)

void deleteProduct(int ID)

product searchProduct(int ID)

}

if (person is client)

{

able to call following functions:

void addProduct(product p1) //This function will be called implicitly when the client returns a product

void deleteProduct(int ID) //This function will be called implicitly when the client checks out a product

product searchProduct(int ID) //This function can be called explicitly by the client

}

The manager class will have a function called addProductToInventory which could look like this:

void addProductToInventory() //Since this function is ONLY in the manager class, a client can’t add a product to the inventory explicitly.

{

cout << “Enter title of product: ” << endl;

cin >> title

…….

…….

…….

product p1 = new product (title, ……)

addProduct(product p1)

}