/\*

char p\_sku[max\_sku\_length + 1];

char p\_unit[max\_name\_length + 1];\*/

char \*p\_name = new char[max\_unit\_length + 1];

char tax;

int p\_current;

int p\_needed;

double single\_Unit;

bool TR;

bool Fl;

if (!is.fail())

{

cout << " Sku: ";

is >> product\_sku;

cin.ignore();

cout << "Name (no spaces):";

is >> p\_name;

name(p\_name);

cout << " Unit: ";

is >> product\_unit;

cout << " Taxed? (y/n): ";

is >> tax;

if (!is.fail())

{

error.clear();

if (tax)

{

TR = tax == 'y' || 'Y';

Fl = tax == 'n' || 'N';

if (TR)

{

Tax = true;

}

if (Fl)

{

Tax = false;

}

if (!(TR || Fl))

{

is.setstate(std::ios::failbit);

error.message("Only (Y)es or (N)o are acceptable");

return is;

}

}

}

else

{

is.setstate(std::ios::failbit);

error.message("Only (Y)es or (N)o are acceptable");

return is;

}

cout << " Price: ";

is >> single\_Unit;

if (is.fail())

{

error.clear();

is.setstate(ios::failbit);

error.message("Invalid Price Entry");

return is;

}

else

price(single\_Unit);

cout << " Quantity on hand: ";

is >> p\_current;

if (is.fail())

{

error.clear();

error.message("Invalid Quantity Entry");

is.setstate(ios::failbit);

return is;

}

else

{

quantity(p\_current);

}

cout << "Quantity needed:";

is >> p\_needed;

cin.ignore();

if (is.fail())

{

error.clear();

error.message("Invalid Quantity Needed Entry");

is.setstate(ios::failbit);

return is;

}

else

{

qtyNeeded(p\_needed);

}

if (!is.fail())

{

error.clear();

}

}

return is;