Exercise 1: Search for H -> gamma gamma

Part 2:

- **Plot the background simulation**
  - `Signal_1fb.root` and `Background_1fb.root` are the signal and background simulations corresponding to 1 inverse fb.
  - Inside the simulation files a TTree is stored called `tree`, which contains two variables `invariantMass and eventWeight`, the invariant diphoton mass and the event weight, respectively.
  - Create histograms and fill them with `invariantMass` weighted by the `eventWeight`.
  - Scale the simulation to the correct integrated luminosity of the data and compare the data to the background-only hypothesis. (Hint: event-by-event loop, SetBranchAddress, etc)

Ok great, but how do we know if we see a signal in the data? We have a background and signal simulation, let's load the files and read in the trees:

In [5]:

```
TFile *fSig = new TFile("Signal_1fb.root",     "READ");
TFile *fBkg = new TFile("Background_1fb.root", "READ");

TTree *tSig = (TTree*) fSig -> Get("tree");
TTree *tBkg = (TTree*) fBkg -> Get("tree");
```

Inside the trees we see two variables:

`invariantMass and eventWeight`

Link the branch variables to some local variables!

In [6]:

```
double mass_sig; double eventWeight_sig;
double mass_bkg; double eventWeight_bkg;

tSig -> SetBranchAddress("invariantMass", &mass_sig);
tSig -> SetBranchAddress("eventWeight",   &eventWeight_sig);

tBkg -> SetBranchAddress("invariantMass", &mass_bkg);
tBkg -> SetBranchAddress("eventWeight",   &eventWeight_bkg);
```

Now define two histograms and fill them with the invariant mass weighted by the event weight. You will need to loop over the trees!

In [7]:

```
// define two histograms
TH1D *hSig = new TH1D("signal", "", 50, 100, 160);
TH1D *hBkg = new TH1D("bkg",    "", 50, 100, 160);

int nEntries_Sig = tSig -> GetEntries();
int nEntries_Bkg = tBkg -> GetEntries();
```

Now loop over the trees:

In [8]:

```
for(int i = 0; i < nEntries_Sig; ++i){
    tSig -> GetEntry(i);
    hSig -> Fill(mass_sig, eventWeight_sig);
}

for(int i = 0; i < nEntries_Bkg; ++i){
    tBkg -> GetEntry(i);
    hBkg -> Fill(mass_bkg, eventWeight_bkg);
}
```

The simulated events correspond to 1 inverse fb, however we have recorded 100 inverse fb, so we need to scale it

In [9]:

```
std::cout << "Before reweighting: " << std::endl;
std::cout << "Signal:\t\t" << hSig -> Integral() << "\n" << "Background:\t" << hBkg -> Integral() << std::endl;

hSig -> Scale(100.0);
hBkg -> Scale(100.0);

std::cout << "After reweighting: " << std::endl;
std::cout << "Signal:\t\t" << hSig -> Integral() << "\n" << "Background:\t" << hBkg -> Integral() << std::endl;
```

```
Before reweighting:
Signal:         16.8333
Background:     6000
After reweighting:
Signal:         1683.33
Background:     600000
```

Ok, great, that makes sense, since we had around 600k data events, lets plot the background model now:

In [10]:

```
hBkg -> SetLineColor(kBlue);
hBkg -> Draw("HSAME");
c ->Draw()
```