PHY213 - KT II
Exercise Sheet 3

Frühjahrssemester 2018
Prof. N. Serra

D. Lancierini

http://www.physik.uzh.ch/de/lehre/PHY213/FS2018.html

Issued: 16.03.2018
Due: 21.03.2018 10:15

Exercise 1: PYTHIA generator for HEP

PYTHIA 8 is a Monte Carlo framework for event generation, it can simulate both lepton and hadron colliders. Some useful documentation can be found in here:

- Particle PDG definition scheme: http://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf.
- Pythia manual: http://home.thep.lu.se/~torbjorn/pythia81html/Welcome.html.
- Pythia tutorial: http://hep.ps.uci.edu/~arajaram/worksheet.pdf.

To provide a first interaction with PYTHIA8 in its standalone running version, start from this simple script below:

```
// example01.cc

#include "Pythia8/Pythia.h" //Includes headers Pythia
using namespace Pythia8; //Let Pythia8:: be implicit



//Begin main program

int main() {

   // Set up generation
   Pythia pythia;

   //Allow no substructure in e+- beams
   pythia.readString("PDF:lepton = off");

   //Process selection

   pythia.readString("WeakSingleBoson:ffbar2gmZ=on");

   //LEP1 initialization at Z0 mass.

   pythia.readString("Beams:idA = 11");
   pythia.readString("Beams:idB = -11");
```

```
   double mZ = pythia.particleData.m0(23);

   pythia.settings.parm("Beams:eCM", 100);
   pythia.init();

   // Generate event(s)
   pythia.next(); // Generate an(other) event. Fill event record
   pythia.stat(); // Extra information
   pythia.event.list(); //Print contents of event record

return 0;

}
```

In order to run it access download the Makefile

https://drive.google.com/open?id=1cmXN-seVdVDWVg4vSHiUjSa2vplkR9vd

After compiling the first script, execute it and analyse the output.

Exercise 2: Event loop

Now we wish to generate more than one event. To do this let's modify the previous script introducing a 5 iterations loop around `pythia.next()` and `pythia.event.list()`. The program will now generate and print 5 events; each call to `pythia.next()` resets the event record and fills it with a new event

a) To obtain statistics on the number of events generated of the different kinds and the estimated cross sections, add

```
pythia.statistics();
```

just before the end of the program

b) In order to avoid major abort problems, in the rare case in which the output of `pythia.next()` is false, it is recommended to nest the event loop in an if statement.

Exercise 3: Particle loop

Looking at the `pythia.event.list()` listing for a few events at the beginning of each run is useful to make sure you are generating the right kind of events, at the right energies, etc. For real analyses, however, you need automated access to the event record.

a) Insert the following loop after `pythia.next()` (but fully enclosed by the event loop)

```
for (int i = 0; i < pythia.event.size(); ++i) {
   cout << "i = " << i
      << ", id = " << pythia.event[i].id() << endl;
}
```

This is said to be a particle loop, inside this loop, you can access the properties of each particle `pythia.event[i]`

b) The method `id()` returns the PDG identity code of a particle. Try to print out a list of the PDG code of every particle in the event record

c) Use the PDG identity code to retrieve the rapidity `eta()` of all muons of the event.