declaration file
`StableNucleus.h`

```
#ifndef STABLENUCLEUS_H
#define STABLENUCLEUS_H

#include "Nucleus.h"

using namespace std ;


class StableNucleus : public Nucleus {

 public:

// costruttori

  StableNucleus() ;

  StableNucleus
       (string Name, double A, int Z) ;


// distruttore

   virtual ~StableNucleus() ;


// metodi di tipo get


// altri metodi

   virtual void print() ;

   virtual int N_nuclei
       (int N0, double t0, double t) ;

   virtual double Activity
       (int N0, double t0, double t) ;



} ;


#endif
```

declaration file
`ActiveNucleus.h`

```
#ifndef ACTIVENUCLEUS_H
#define ACTIVENUCLEUS_H

#include "Nucleus.h"

using namespace std ;


class ActiveNucleus : public Nucleus {

 protected:

  double tau ;

 public:

// costruttori

  ActiveNucleus() ;

  ActiveNucleus
       (string Name, double A, int Z) ;

  ActiveNucleus
   (string Name, double A, int Z, double Tau) ;


// distruttore

   virtual ~ActiveNucleus() ;


// altri metodi

   virtual void print() ;

   virtual int N_nuclei
       (int N0, double t0, double t) ;

   virtual double Activity
       (int N0, double t0, double t) ;


} ;


#endif
```

UNIVERSITA' DEGLI STUDI ROMA TRE
DIPARTIMENTO DI MATEMATICA E FISICA

Corso di Laboratorio di Calcolo (Parte A)
Esercitazione Individuale

Nuclei Stabili e Nuclei Radioattivi

gennaio 2014

1) Costruire una classe (che potrete chiamare ad esempio *`Nucleus`*) per descrivere le caratteristiche di una sostanza di nome `name`, di peso atomico A e di numero atomico Z.

In particolare, dovranno essere definiti ed implementati i seguenti metodi

i) Costruttore di default (*`Nucleus()`*) , Costruttore con parametri (*`Nucleus`*`(string name, int A, int Z)`) e distruttore (*`~Nucleus()`*).

ii) Metodo di tipo Get che forniscano il nome del nucleo (*`name()`*), il peso atomico (*`A()`*) ed il numero atomico (*`Z()`*).

iii) Un metodo di tipo print (*`print()`*) che permetta di visualizzare sullo schermo il nome del nucleo, il suo peso atomico ed il suo numero atomico.

Si costruisca un semplice programma main per controllare la funzionalita' dei metodi implementati.

(5 punti)

2) Costruire due nuove classi (che potrete chiamare ad esempio *`StableNucleus`* e *`ActiveNucleus`*), che ereditano dalla classe *`Nucleus`* definita in precedenza. In particolare:

i) La classe `ActiveNucleus` sia caratterizzata da un ulteriore attributo costituito dal tempo di decadimento τ.

ii) Si implementino, per ciascuna classe, tutti i Costruttori del tipo definito al punto 1) e, per la classe *`ActiveNucleus`*, il Costruttore con tutti i parametri (*`Nucleus`*`(string name, int A, int Z, double tau)`). Per entrambe le classi si implementino i distruttori.

iii) Si implementi, nelle due classi, il metodo *`print()`* , in modo che visualizzi sullo schermo anche il valore del tempo di decadimento oppure la scritta "Nucleo Stabile"

iv) Si renda virtuale il metodo *`print()`*

Si modifichi il programma main di test per verificare la correttezza dei metodi implementati **(3 punti)**

3) Si implementino due metodi virtuali (*int N_nuclei(int N0, double t0, double t)* e *double Activity(int N0, double t0, double t)*) che forniscano il numero di nuclei residui e l'attivita' (in decadimenti al secondo) al tempo t, dato un numero iniziale di nuclei N_0 al tempo t_0.

Si modifichi il programma main di test per verificare la correttezza dei metodi implementati, per diversi valori del tempo t. **(2 punti)**

========

Suggerimento

i) Per un nucleo stabile, il metodo *int N_nuclei* dovra' restituire sempre il valore N0 e il metodo *Activity* dovra' restituire sempre il valore 0;

i) Per un nucleo instabile, una volta implementato il il metodo *int N_nuclei* (utilizzando la relazione riportata in Appendice), il metodo *Activity* potra' essere semplicemente implementato nella forma:

```
double ActiveNucleus::Activity(int N0, double t0, double t) {
            return (this->N_nuclei(N0,t0,t))/tau ;
    }
```

APPENDICE

Valori delle costanti di decadimento per alcuni nuclei radioattivi β^-

Cesio	Cs	A=135	Z=55	$\tau = 30.2$ y
Stronzio	Sr	A=90	Z=38	$\tau = 28.5$ y
Cobalto	Co	A=60	Z=27	$\tau = 5.27$ y
Rutheford.	Ru	A=106	Z=44	$\tau = 1.02$ y

Relazioni Fondamentali

$$N = N_0 e^{-\frac{(t-t_0)}{\tau}} \qquad \left|\frac{dN}{dt}\right| = \frac{N}{\tau}$$

N.B. Tempo per l'esercitazione: 3 ore (14.30-17.30)

Si possono usare appunti delle lezioni e manuali di C++

La prova d'esonero si intende superata se il programma funziona (compilazione, linking, esecuzione)

declaration file `Nucleus.h`

```
#ifndef NUCLEUS_H
#define NUCLEUS_H

#include <iostream>
#include <string>
#include <cmath>

using namespace std ;


class Nucleus {

 protected:

  string name ;
  double An   ;
  int    Zn   ;

 public:


 // costruttori

    Nucleus() ;

    Nucleus (string Name, double A, int Z) ;


 // distruttore

    virtual ~Nucleus() ;


 // metodi di tipo Get

    string Name() ;
    double A()    ;
    int Z()       ;


 // altri metodi

    virtual void print() ;

    virtual int N_nuclei(int N0, double t0, double t)    ;

    virtual double Activity(int N0, double t0, double t) ;


} ;


#endif
```