Esame di Laboratorio di Fisica Computazionale

21 luglio 2017, ore 9.30

1 Mathematica

1. Si definisca una funzione `myEigenvalues` che prende in input una generica matrice M e restituisce in output la lista dei suoi autovalori.
   (Si possono usare i comandi nativi di `Mathematica` *con l'esclusione di* `Eigenvalues`.)

2. Si definisca una funzione `myDet` che prende in input una generica matrice M di dimensioni $n \times n$ e restituisce in output il determinante della matrice calcolato secondo la formula

$$\det(M) \equiv \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} M_{i,\sigma(i)} \tag{1}$$

   dove S_n sono tutte le permutazioni della lista di interi $\{1, 2, 3, \ldots, n\}$ mentre $\mathrm{sgn}(\sigma)$ è il segno della permutazione.
   (Si possono usare i comandi nativi di `Mathematica` *con l'esclusione di* `Det`.)

3. Si calcoli la soluzione generale dell'equazione differenziale

$$y'(t) + 4t\, y(t) = 2t \tag{2}$$

4. Si definisca con l'ausilio del comando `Module` una funzione `myDSolve` che calcola la soluzione generale di un'equazione differenziale lineare del primo ordine, implementando il metodo di Eulero della variazione delle costanti. In particolare, se l'equazione ha la forma

$$y'(t) + a(t)\, y(t) = f(t) \tag{3}$$

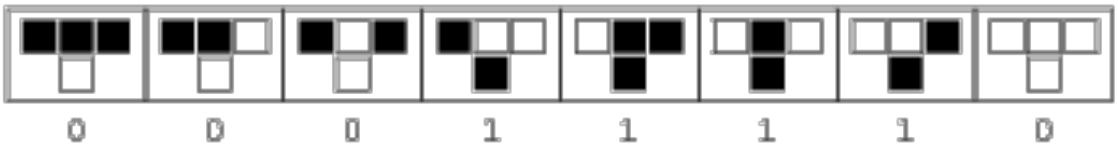
   allora si dimostra che la soluzione può essere scritta come

$$\tilde{y} = e^{-A(t)} \int dt\, e^{+A(t)}\, f(t) \tag{4}$$

   dove $A(t)$ è una primitiva di $a(t)$.
   La funzione `myDSolve` deve avere tre argomenti: l'equazione differenziale, la funzione incognita, la variabile rispetto a cui si integra l'equazione.
   Si verifichi il funzionamento nel caso specifico di equazione 2.
   Suggerimento: si consideri l'utilizzo del comando `Coefficient` per ricavare l'espressione di $a(t)$.
   Suggerimento: si tenga presente che `Mathematica` non mette il termine costante quando calcola un integrale indefinito.

5. Automa cellulare

- Si scriva una funzione che inizializza una lista contenente $2n+1$ elementi, tutti zero con l'eccezione dell'elemento $n+1$, che deve essere posto pari a 1.
  (Si consideri l'utilizzo combinato di `ReplacePart` *e di* `Table`*).*
- Si scriva una lista di regole di sostituzione che realizzi le trasformazioni rappresentate graficamente in figura.
  (Si consideri per la singola sostituzione una regola del tipo $\{a, b, c\} \to d$*)*



- La trasformazione di una lista esistente in una nuova, secondo il meccanismo dell'automa cellulare, richiede di applicare a ciascuna terna di elementi consecutivi della lista il set di regole di sostituzione del punto precedente. Per semplicità, il primo e l'ultimo elemento della lista possono essere ricopiati da una riga all'altra, mentre si devono calcolare tutti gli elementi nuovi dal secondo al penultimo.
  Si utilizzino i comandi `Table` *per generare i nuovi elementi, dal secondo al penultimo, e poi* `Prepend` *e* `Append` *per ricopiare il primo e l'ultimo.*

2 C++: Esercizio

Si risolva l'esercizio proposto. Per facilitare la correzione, se possibile includere tutto in un unico file sorgente. Orientativamente, la sufficienza è raggiunta risolvendo correttamente i primi tre punti.

Esercizio

Si vuole scrivere un semplice framework per gestire pedaggi autostradali.

1. Si scriva una classe `Vehicle`, che rappresenterà un veicolo. Si mettano tra i membri *private* la velocità massima consentita al veicolo (in km/h) e il pedaggio da pagare per chilometro percorso (in €/km). Si scriva un opportuno costruttore che inizializzi queste ultime due variabili, con *valori di default* 130 e 0.1. Attenzione: si vuole che il costruttore, se chiamato con un solo parametro, imposti il pedaggio e assegni il valore di default alla velocità.

2. Si aggiunga tra i membri *private* la targa del veicolo (una variabile di tipo `std::string`, che è un container di oggetti di tipo `char`, e ha un'interfaccia compatibile con gli altri container STL). Il costruttore dovrà inizializzarla ad una stringa di 5 caratteri maiuscoli casuali (si veda `www.asciitable.com` per una tabella dei valori numerici ASCII).

3. Si scriva il costruttore di copie di `Vehicle`, usando la sintassi della lista di inizializzazione.

4. Si scriva una funzione pubblica `print` che stampi la targa su `std::cout`. Si scriva inoltre una *access function* che restituisca *per valore* la targa. Rispondere brevemente (in un commento): dall'esterno della classe `Vehicle` è possibile usare questa funzione per modificare la targa di un veicolo? Se sì dire come; altrimenti dire come modificare il codice in modo che diventi possibile.

5. Si scriva una funzione `toll` che prenda come argomenti la distanza percorsa dal veicolo (in km) e il tempo impiegato a percorrerla (in ore): dovrà restituire il pedaggio complessivo da pagare (in €), più una eventuale multa di 50 € nel caso in cui la velocità media sia stata superiore a quella massima consentita. Si vuole che questa funzione abbia comportamento polimorfico. Bonus: si scriva questa funzione usando l'operatore ternario "?:".

6. Si scriva una classe `Pullman`, derivata di `Vehicle`, il cui pedaggio sarà fissato a 0.2 €/km e la velocità massima a quella di default. Questo tipo di veicolo possiede una velocità "consigliata", stando sotto la quale il pedaggio (come calcolato dalla funzione della classe base) viene scontato del 20%. Si aggiunga questa velocità ai membri privati e si scriva un opportuno costruttore che la inizializzi. Si scriva anche l'*override* della funzione `toll`, riutilizzando per quanto possibile la funzionalità già implementata in `Vehicle`.

7. Per controllare il comportamento polimorfico, nel *main* si istanzi un `std::vector` che contenga un `Vehicle` (inizializzato coi valori di default) e un `Pullman` con velocità consigliata 100 km/h, allocati dinamicamente [si consideri se usare valori, puntatori o referenze]. Per ogni elemento del vettore si stampi su `cout` il pedaggio che il veicolo deve pagare se ha percorso 90 km in un'ora. [Ci si ricordi di deallocare gli oggetti allocati dinamicamente.]

8. Si scriva una classe `Highway`, che si occuperà di registrare i veicoli in entrata e in uscita, e di calcolare per ognuno il pedaggio da pagare. Si vuole mettere, tra i membri privati, un container che tenga in memoria, per ogni veicolo in ingresso, la sua targa associata all'orario di ingresso (per semplicità si usino variabili `double` per l'orario misurato in ore, come fatto sopra).

   [Il container `std::map` della STL è utile a questo scopo: un `std::map<T,U>` associa un oggetto di tipo `U` a un oggetto di tipo `T` (quest'ultimo detto *chiave*); sarà necessario usare soltanto l'operatore "[]" di `set::map`, che restituisce per referenza l'oggetto associato alla chiave passata come argomento, o crea una nuova coppia chiave-valore nel caso la chiave non esista.]

9. Tra i membri pubblici di `Highway`, si scrivano le funzioni `enter` e `exit`, alle quali vengono passati un veicolo e un tempo di ingresso (per `enter`) o uscita (per `exit`). La funzione `exit` dovrà restituire il pedaggio da pagare (per semplicità fissiamo la lunghezza del tratto di autostrada a 90 km).

10. Nel `main`, istanziare un oggetto di tipo `Highway` e registrare l'ingresso dei due veicoli inseriti nel vettore al punto 7, il primo al tempo 0 h, il secondo al tempo 0.3 h. Infine registrare l'uscita dei due veicoli (entrambi esattamente 1 h dopo l'ingresso) e stampare su `cout` i pedaggi calcolati dallo `Highway`.

3 Shell scripting

1. Si scriva uno script che scambia le due colonne del file `prova.txt` e salva il risultato nel file `scambiate.txt`.

2. Si scriva uno script che calcola la media dei valori della terza colonna del file `voti.txt`

3. Si scriva uno script che seleziona, tra tutti i files presenti nella directory `/home/vicini/corso1617/`, quelli creati nel mese di aprile 2017 e li scrive nel file `aprile.txt`.