Laboratorio di Calcolo per Fisici, Esercitazione Valutata

Nome _____ Cognome _____

Matricola _____ ☐ Ritirato/a

Lo scopo di questa Esercitazione valutata è di scrivere un programma in C e uno script in python seguendo la traccia riportata di seguito. Si tenga presente che:

1. Per svolgere la prova si hanno 3 ore a disposizione.
2. Si possono usare solo libri di testo, prontuari e gli appunti scritti di pugno.
3. Non è ammesso parlare con nessuno nè utilizzare cellulari, tablet o laptop, pena l'annullamento del compito.
4. Il programma va scritto e salvato esclusivamente sul computer del laboratorio, a cui si deve accedere utilizzando come username `studente` e password `informatica`.
5. **Tutti i file vanno salvati in una cartella chiamata `LCNG-NOME-COGNOME` nella home directory**, dove `NOME` e `COGNOME` sono il tuo nome e cognome rispettivamente. Ad esempio se ti chiami *Marco Rossi* devi creare una cartella chiamata `LCNG-MARCO-ROSSI`. Se invece il tuo nome e cognome sono *Paolo De Marchis* allora devi creare una cartella chiamata `LCNG-PAOLO-DE-MARCHIS`, contente tutti i file specificati nel testo. **Tutto ciò che non si trova all'interno della cartella suddetta non verrà valutato.** In tutti i programmi che scriverai inserisci all'inizio sempre un commento con il tuo nome, cognome e numero di matricola.
6. **Il presente testo va consegnato indicando nome, cognome e numero di matricola** (vedi sopra), barrando la casella "Ritirato" se ci si vuole ritirare, ovvero se non si vuole che la presente prova venga valutata. I codici consegnati che presentano errori in fase di compilazione non verranno valutati sufficienti.

Gli algoritmi genetici cercano di simulare il processo di evoluzione di una specie. L'algoritmo prevede di creare un gruppo di organismi che hanno ciascuno un numero fissato di geni (di solito scelti a caso). Questi organismi rappresentano la "generazione madre". Poi si valuta e si classifica l'idoneità genetica (*fitness*) di ciascuno degli organismi rispetto ad una sequenza genetica di riferimento che rappresenta "l'organismo perfetto" in base ad alcuni criteri specifici (come ad esempio, una maggiore capacità di sopravvivere in condizioni estreme). Successivamente si scelgono alcuni degli organismi più idonei e li si lascia riprodurre tra loro per dare vita alla "generazione figlia". Una volta che la generazione madre si è riprodotta tutti i suoi membri "muoiono" e la generazione figlia diventa la nuova generazione madre. Si ripetono quindi i passi dell'analisi genetica e della riproduzione finché non si siano succedute M generazioni.

Scrivere un codice C chiamato `genetica.c` che simuli l'evoluzione genetica descritta in precedenza. In particolare:

- ogni generazione è composta da `N`=200 organismi, ciascuno dei quali ha un corredo genetico costituito da `NGENI`=18 geni. Per semplicità ogni gene verrà rappresentato da uno dei valori 0, 1, 2 o 3. `N` e `NGENI` sono costanti simboliche passate al preprocessore con la direttiva `#define`.

- il codice genetico della prima generazione di organismi viene memorizzato all'interno della matrice `current_generation[][]` di dimensioni opportune. La sequenza genetica di ciascun organismo si ottiene dall'estrazione casuale di un numero intero compreso tra 0 e 3 (estremi inclusi) per `NGENI` volte, memorizzando ogni valore estratto all'interno della matrice. La procedura per generare il codice genetico di tutti gli organismi deve avvenire all'interno della funzione `GeneraDNA`(), che prende in input la matrice `current_generation[][]` e la riempie come descritto sopra. La funzione deve anche popolare con le stesse regole un array passato in input e chiamato `modeldna[]` che conterrà la sequenza genetica di riferimento dell'organismo perfetto. La funzione viene chiamata una volta sola nel `main` per inizializzare la prima generazione di organismi.

- All'interno di una funzione chiamata `Fitness()` il codice genetico di ciascun organismo della generazione madre viene confrontato con quello contenuto in `modeldna[]`. Ogni volta che il gene i-esimo di un organismo corrisponde a quello i-esimo di `modeldna[]` viene incrementato un contatore, inizialmente posto pari a zero. I contatori di tutti gli organismi devono essere contenuti in un array chiamato `contageni[]`, passato in input alla funzione e che va inizializzato ogni volta prima di effettuare il confronto con il codice genetico di riferimento. Infine, la funzione restituisce la somma dei valori contenuti in `contageni[]` che servirà per selezionare gli organismi per la riproduzione e che chiameremo `sum_geni`.

- Gli organismi con un corredo genetico più simile a quello di riferimento avranno una probabilità maggiore di riprodursi. Per estrarre con maggior probabilità uno di questi organismi, si crei una funzione chiamata `SelezionaGenitore()` che restituisce l'indice associato all'organismo da far riprodurre. La funzione prende in input `contageni[]` e `sum_geni` ed esegue i seguenti passi. Prima di tutto estrae un numero casuale razionale `y` tra $[0, \texttt{sum\_geni})$. Successivamente, all'interno di un ciclo iterativo, effettua l'operazione `sum+=contageni[i]`, dove `i` è l'indice del ciclo e `sum` è opportunamente inizializzata. Il ciclo si interrompe se `sum>=y` e la funzione restituisce il valore del passo a cui si è verificata la condizione.

- Per dare vita alla generazione figlia si deve creare una funzione chiamata `Riproduzione()`. In questa funzione il codice genetico di `N` nuovi individui verrà creato e memorizzato in una matrice chiamata `next_generation[][]` secondo il seguente schema. Per ciascun nuovo organismo si selezionano due genitori chiamando due volte la funzione `SelezionaGenitore()`. Il codice genetico del figlio sarà costituito dai primi `x` geni del primo genitore e dai restanti geni del secondo genitore, dove `x` è un numero casuale intero scelto tra [0,NGENI). Infine ogni volta che una generazione figlia viene creata, questa diventa la nuova generazione madre.

- Nel `main()` le funzioni `Fitness()` e `Riproduzione()` sono racchiuse in un ciclo iterativo sul numero massimo `M=20` di generazioni che si possono creare. Ad ogni passo del ciclo vengono stampate su un file chiamato `fitness.dat` due colonne: la prima è l'indice della generazione di organismi (1,2,3....), mentre la seconda è la differenza tra la fitness ideale (pari a `NGENI`) e il valor medio del contenuto di `contageni[]`, ossia il valor medio della fitness di quella generazione. Gli interi vanno stampati con almeno 4 cifre, mentre i numeri razionali con almeno 7 di cui 5 dopo la virgola.

- Creare uno script python chiamato `fitness.py` che grafichi il contenuto di fitness.dat. Controllare che la curva, in scala semilogaritmica, sia approssimativamente una retta con coefficiente angolare negativo. Per settare la scala logaritmica per l'asse y usare il comando python `plt.yscale('log')`. Corredare il grafico di titolo, nome sugli assi e legenda e salvarlo in `fitness.png`.