

Laboratorio di Calcolo, Esame del 06/02/2026

Anno accademico 2025-26

Nome: _____ Cognome: _____
Matricola: _____ Ritirata/o

Lo scopo di questa prova d'esame 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 il compito si hanno a disposizione 3 ore.
2. Si possono usare libri di testo, proutuari e i propri appunti cartacei scritti a mano ma non è ammesso parlare con nessuno né utilizzare cellulari, tablet o laptop, pena l'annullamento del compito.
3. **Tutti i file vanno salvati in una cartella chiamata LCFEB26_NOME_COGNOME nella home directory**, dove NOME e COGNOME indicano rispettivamente il vostro nome e cognome. Ad esempio lo studente *Nicolò Maria De Rossi Salò* deve creare una cartella chiamata LCFEB26_NICOLOMARIA_DEROSSISALO contenente tutti i file specificati nel testo. **Tutto ciò che non si trova all'interno della cartella suddetta non verrà valutato.** In tutti i programmi e script inserite all'inizio un commento con il vostro nome, cognome e numero di matricola.
4. **Dovete consegnare il presente testo indicando nome, cognome e numero di matricola** (vedi sopra), barrando la casella "Ritirato/a" se ci si vuole ritirare, ovvero se non si vuole che la presente prova venga valutata.
5. **Per consegnare il compito** dovreste eseguire, all'interno della cartella creata in precedenza (come spiegato al punto 3), il seguente comando da terminale: `cp * /media/sf_esame/`

Lo scopo di questa prova d'esame è di scrivere un programma in linguaggio C che simuli una versione semplificata del gioco *snake*, descrivendo la crescita progressiva di un "serpente" discreto su una griglia bidimensionale.

Un serpente è rappresentato da una sequenza ordinata di coordinate intere relative alle caselle occupate su una griglia bidimensionale illimitata. In linguaggio C tale configurazione viene descritta tramite un array bidimensionale `pos`, in cui `pos[i][0]` e `pos[i][1]` rappresentano le coordinate intere x_i e y_i della casella occupata dal segmento i . Con questa definizione, $i = 0$ corrisponde al primo segmento del serpente, mentre $i = l - 1$ corrisponde all'ultimo segmento di un serpente di lunghezza l .

Il serpente non si muove come nel gioco classico, ma cresce progressivamente: ad ogni passo temporale viene aggiunto un nuovo segmento in una posizione adiacente all'ultimo segmento presente. La direzione di crescita viene scelta casualmente tra le quattro direzioni possibili sulla griglia (destra, sinistra, alto, basso), evitando di "tornare sui propri passi" (si veda il pannello (a) della figura).

La simulazione termina quando il nuovo segmento aggiunto si sovrappone a uno dei segmenti precedenti (auto-intersezione, si veda il pannello (b) della figura), oppure quando il serpente ha raggiunto una certa lunghezza massima.

Esercizio in C

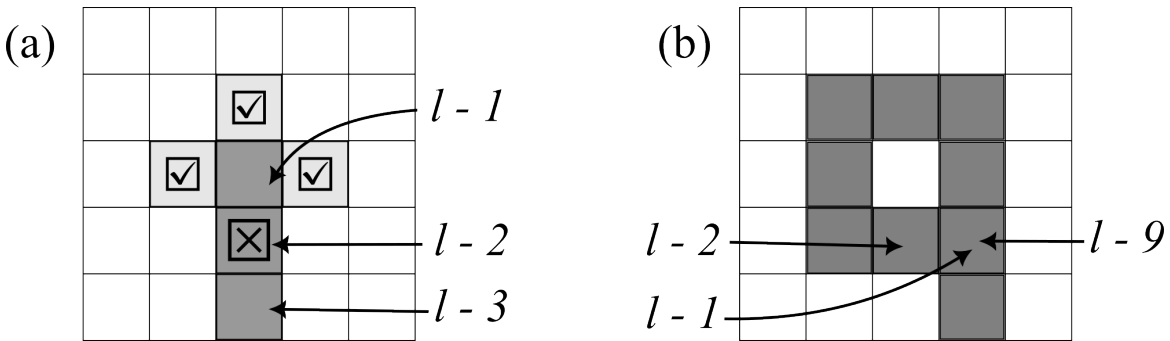


Figura 1: (a) Possibili posizioni valide (✓) e non valide (⊗) per il segmento l -esimo. (b) Un esempio di serpente di lunghezza l che si auto-interseca: l'ultimo segmento, di indice $(l-1)$, ha le stesse coordinate del segmento $(l-9)$ -esimo.

Scrivere un programma in linguaggio C chiamato `NOME_COGNOME.c` che simuli la crescita di un serpente su una griglia bidimensionale.

In particolare, il programma dovrà:

1. Definire tramite opportune direttive `#define` le seguenti costanti simboliche:
 - `NT = 50000`, numero di simulazioni indipendenti,
 - `LINIT = 10`, lunghezza iniziale del serpente,
 - `LMAX = 500`, lunghezza massima del serpente.
2. Utilizzare il generatore di numeri casuali `drand48()` usando come *seed* il valore 131.
3. Dichiarare nel `main()` un array bidimensionale `pos` di tipo e lunghezza opportuni, in cui verranno memorizzate le coordinate dei segmenti del serpente.
4. Inizializzare la configurazione iniziale del serpente tramite una funzione `init_snake()`, che disponga i primi `LINIT` segmenti lungo una linea retta, in modo che il segmento i -esimo abbia coordinate $(0, i)$.
5. Implementare direttamente nel `main()` un ciclo iterativo che, ad ogni passo, aggiunga un nuovo segmento tramite una funzione `add_segment()`.
6. Concludere la simulazione quando il serpente si auto-interseca, ovvero quando l'ultimo segmento aggiunto si sovrappone a uno dei segmenti precedenti, oppure quando viene raggiunto il numero massimo di segmenti `LMAX`.
7. Dopo ogni simulazione aggiornare un istogramma delle lunghezze finali del serpente. Per fare ciò si utilizzi un array chiamato `histo` di lunghezza opportuna, tale che `histo[1]` contenga il numero di volte in cui la simulazione è terminata con una lunghezza 1 del serpente.
8. Salvare in un file chiamato `isto_lunghezza.dat` le frequenze delle lunghezze, ottenute normalizzando l'istogramma, con 6 cifre significative.

Nello scrivere il programma si dovranno definire almeno le seguenti funzioni:

1. Una funzione `init_snake()`, che prenda come argomento l'array `pos` e la lunghezza l , passata tramite un puntatore (ovvero *by reference*), e inizializzi la configurazione iniziale del serpente e la lunghezza l .
2. Una funzione chiamata `add_segment()`, che prenda come argomento l'array `pos` e la lunghezza corrente l del serpente, passata tramite un puntatore (*by reference*), e aggiunga il segmento l -esimo scegliendo casualmente una delle quattro direzioni possibili, aggiornando anche il valore di l .

In particolare, se l'ultimo segmento ha coordinate (x_{l-1}, y_{l-1}) , allora il nuovo segmento potrà essere posto in una delle seguenti posizioni (scelta casualmente):

$$(x_{l-1} + 1, y_{l-1}), \quad (x_{l-1} - 1, y_{l-1}), \quad (x_{l-1}, y_{l-1} + 1), \quad (x_{l-1}, y_{l-1} - 1).$$

Tuttavia, la funzione deve assicurarsi che la crescita non avvenga immediatamente nella direzione che riporterebbe il serpente sulla casella occupata dal segmento $(l - 2)$ -esimo. In altre parole, tra le quattro direzioni possibili, deve essere esclusa quella che produrrebbe una sovrapposizione con il segmento precedente di due passi (come mostrato nella figura).

3. Una funzione `check_overlap()`, che controlli se l'ultimo segmento aggiunto coincide con uno dei segmenti precedenti, restituendo 1 in caso affermativo e 0 altrimenti. Questa funzione prenderà come argomenti l'array `pos` e la lunghezza attuale del serpente l .

► **Esercizio in Python** Dopo aver verificato il corretto funzionamento del programma in C e averlo eseguito, scrivere uno script Python `NOME_COGNOME.py` che legga i dati del file `isto_lunghezza.dat` e li riporti in un grafico completo di titolo ed etichette degli assi. Il grafico va salvato nel file `NOME_COGNOME.png`.