Laboratorio di Calcolo, Canale Pet-Z
Prova di esame, appello Febbraio 2026

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

1. Il tempo a disposizione è di 3 ore. Sono ammessi libri di testo, prontuari, appunti. Non si può parlare con nessuno, utilizzare cellulari/tablet/laptop, pena l'annullamento del compito.
2. Il programma va scritto e salvato esclusivamente sul computer del laboratorio, a cui si deve accedere utilizzando come username **studente** e come password **informatica**.
3. **Tutti i file vanno salvati in una cartella chiamata `LCSR_COGNOME_NOME` nella home directory**, dove `NOME` e `COGNOME` indicano rispettivamente il tuo nome e cognome. Ad esempio lo studente *Nicolò De Rossi* deve creare una cartella chiamata `LCSR_DEROSSI_NICOLO` contenente tutti i file specificati nel testo. **Tutto ciò che non si trova all'interno della cartella suddetta non verrà valutato.**
4. Consegnare il presente testo indicando nome, cognome e numero di matricola (vedi sopra), barrando la casella "Ritirata/o" se ci si vuole ritirare, ovvero se non si vuole che la presente prova venga valutata.

Gioco dell'Oca
Il gioco dell'oca è un gioco da tavolo per 2 o più giocatori. Si lancia il dado e si avanza lungo un percorso a spirale con varie caselle speciali. L'obiettivo è arrivare esattamente all'ultima casella.
Alcune caselle aiutano (per esempio le oche ti fanno avanzare di nuovo), altre penalizzano (pozzi, prigioni o locande ti fanno fermare o tornare indietro). Vince chi raggiunge per primo la casella finale rispettando le regole dei tiri.

► **Esercizio in `C`:**

Si vuole realizzare un programma `cognome_nome.c` in linguaggio `C` che simuli il gioco dell'oca che si svolge su un tabellone lineare costituito da 64 caselle numerate da 0 a 63 (inclusi). Tutti gli N giocatori partono dalla casella 0 e il gioco termina quando il primo giocatore raggiunge o supera la casella finale.
Il programma deve svolgere le seguenti operazioni:

1. Tramite direttive del precompilatore, definire il numero `ULTIMACELLA` e il numero massimo di giocatori `MAXGIOCATORI` da utilizzare nel resto del programma.
2. Richiedere all'utente il numero di giocatori, un valore intero compreso tra 2 e 6 (estremi inclusi).
3. Inizializzare un array `posizione[]` di lunghezza opportuna per memorizzare la posizione corrente dei giocatori.
4. Inizializzare un array `fermo[]` di lunghezza opportuna per indicare se un giocatore deve saltare il turno successivo (`0` = può giocare, `1` = fermo).
5. Simulare il gioco tramite un ciclo sui turni. In ogni turno tutti i giocatori giocano in ordine.

(a) Per ciascun giocatore, in base al valore immagazzinato in `fermo[]` si decide se il giocatore salta il turno e il valore in `fermo[]` viene riportato a 0; altrimenti viene lanciato un dado tramite una funzione `dado()` (di tipo e argomenti opportuni) che restituisce un numero intero compreso tra 1 e 6 (estremi inclusi), e la posizione del giocatore viene aggiornata. Utilizzare la funzione `drand48()` (che genera numeri razionali nell'intervallo $[0, 1)$) all'interno di `dado()` per generare le facce.

(b) Dopo il lancio del dado viene chiamata una funzione `controllo()`, di tipo e opportuni argomenti, che verifica se la nuova posizione è una casella speciale e restituisce il numero di posizioni da avanzare o indietreggiare:

- **Caselle con l'oca**: se la posizione è un multiplo di 7, il giocatore avanza di 3 caselle;
- **Labirinto**: se la posizione è un multiplo di 10, il giocatore arretra di 2 caselle;
- **Prigione**: se la posizione è 13, 17 oppure 29, il giocatore dovrà saltare il turno successivo (senza ulteriori spostamenti).

(c) L'array `posizione[]` viene aggiornato in base al valore restituito da `controllo()`

6. Al termine di un turno completo il programma deve scrivere su un file di testo `posizione.txt` una riga di dati contenente il numero del turno seguito dalle posizioni correnti di tutti e N i giocatori, come numeri interi separati da spazi, nel formato:

   ```
   turno pos_G1 pos_G2 pos_G3 ... pos_GN
   ```

7. Il gioco termina quando un giocatore raggiunge o supera la casella finale;

8. Scrivere sullo schermo il numero del giocatore vincitore e il numero totale di turni effettuati.

   ```
   Vincitore: giocatore 3 dopo 10 turni
   ```

9. Determinare quale giocatore ha subito il maggior numero di penalità (prigione o labirinto) stampando un messaggio nel formato seguente:

   ```
   Giocatore 1 con 5 penalita
   ```

   Suggerimento: questo richiede di modificare l'interfaccia della funzione `controllo()` e l'utilizzo di un array `penalty[]`.

10. Girare il programma con 5 giocatori per salvare i dati da usare nella parte successiva.

► **Esercizio in `Python`:**

Scrivete uno script Python `cognome_nome.py` che utilizzi i dati contenuti nel file `posizione.txt` per fare un grafico che mostri la posizione degli N giocatori con colori diversi, in funzione del turno di gioco sull'asse x. Lo script deve salvare il grafico, che dovrà contenere una legenda e opportuni label sugli assi, nel file `oca.png`.