Nome e Cognome _____ Matricola _____

Corso di *High Performance Computing*
Ingegneria e Scienze Informatiche—Università di Bologna

Demo prova scritta

- La prova dura 60 minuti
- Durante la prova non è consentito consultare libri, appunti o altro materiale.
- Non è consentito l'uso di dispositivi elettronici (ad esempio, cellulari, tablet...), né interagire in alcun modo con gli altri studenti pena l'esclusione dalla prova, che potrà avvenire anche dopo il termine della stessa.
- Le risposte devono essere scritte **a penna** su questi fogli, in modo **leggibile**. Le parti illeggibili o scritte a matita saranno ignorate.
- Eventuali altri fogli possono essere utilizzati per la brutta copia ma non verranno valutati.
- I voti saranno pubblicati su AlmaEsami e ne verrà data comunicazione all'indirizzo mail di Ateneo (@studio.unibo.it).
- I voti restano validi fino alla sessione d'esame di **settembre xxx** inclusa. Dopo tale data tutti i voti in sospeso saranno persi.

NON SCRIVERE NELLA TABELLA SOTTOSTANTE

D. 1	D. 2	D. 3	D. 4
/ 8	/ 8	/ 8	/ 8

Domanda 1. Descrivere i concetti di *speedup* ed *efficienza* di applicazioni parallele.

Domanda 2. Si consideri il seguente frammento di codice C che calcola le forze che agiscono su N particelle causate dalla mutua attrazione gravitazionale. La funzione gravity(i, j) (non dettagliata) calcola la forza che agisce tra le particelle i e j. La funzione è simmetrica, cioè gravity(i, j) = gravity(j, i), per cui basta calcolare le forze che agiscono tra le particelle i, j con $i < j$. Si assuma che il valore restituito dalla funzione gravity(i, j) dipenda solo da i e j, e da nessun'altra proprietà delle particelle (in particolare, non dipende dal valore del campo `force`).

```
struct Particle {
      float force; /* la forza è rappresentata da un singolo valore reale */
      /* altri attributi della particella, non mostrati */
};

Particle particles[N];
int i, j;

...

for (i=0; i<N; i++) {
      for (j=i+1; j<N; j++) {
            const float f = gravity(i, j);
            particles[i].force += f;
            particles[j].force += f;
      }
}
```

Discutere i problemi e le possibili soluzioni legate all'uso di OpenMP per parallelizzare il codice precedente.

Domanda 3. Scegliere due delle seguenti primitive MPI di comunicazione collettiva:

- **`MPI_Scatter`**
- **`MPI_Scan`**
- **`MPI_Reduce`**

Per ciascuna delle primitive scelte:

1. Si descriva in modo sintetico ma il più possibile esauriente che cosa fa (non è necessario ricordarsi la sintassi); se lo si desidera si può accompagnare la spiegazione con schemi e diagrammi.
2. Si illustri un caso concreto in cui tale primitiva può essere impiegata; è possibile ricorrere ad esempi visti in aula e/o in laboratorio.

Domanda 4. Si illustrino i concetti di *thread* e *thread block* delle architetture CUDA.