ESERCIZIO 1

Ciccio Pasticcio ha avviato una nuova attività, una libreria per bambini dal nome “Il mondo dei Piccoli”. Per la sua libreria non ha badato a spese, ha comprato tantissimi libri, per bambini di tutte le età. Gli affari però non vanno molto bene; Ciccio è un gran pasticcione e finora non è stato in grado di gestire la libreria. Anzitutto, i tantissimi libri sono tenuti nel più completo disordine e perciò ogni qual volta un cliente fa una richiesta specifica (ad esempio, la favola di Biancaneve) Ciccio impiega tantissimo tempo per trovarla (facendo spazientire il cliente) e spesso non riesce proprio a trovarla. Inoltre, non è in grado di dare consigli ai clienti: ad una mamma che cercava una bella favola da raccontare al suo bambino di due anni per aiutarlo a addormentarsi ha consigliato uno dei libri di Harry Potter ecc. Insomma Ciccio Pasticcio è proprio nei guai. Aiutatelo allora nella gestione del suo negozio.

Esercizio 1.1
A tale scopo, implementare una classe LIBRO i cui dati siano (almeno) il nome del libro, il nome dell’autore, e l’età dei bambini a cui è adatto (ad esempio, se per un libro il campo età contiene 6 allora quel libro è adatto a bambini di 6 anni o più grandi, ma non a bambini più piccoli).

Esercizio 1.2
Implementare una classe libreria che contenga i libri disposti secondo fasce di età e per ogni fascia ordinati in ordine alfabetico. Le fasce di età sono 3, da 1 a 5 anni, da 6 a 10, maggiori di 10.
La classe deve contenere oltre a eventuali costruttori e funzioni di utilità:

- un metodo che ricevuto come parametro il nome di un libro, verifichi se tale libro è presente nella libreria (in una qualsiasi fascia di età)
- un metodo che ricevuto come parametro l’età di un bambino scelga a caso un libro adatto a lui.
- un metodo che inserisca un dato libro nella libreria facendo attenzione all’età e a rispettare l’ordine alfabetico.
- un metodo che elimini un libro dalla libreria.

Esercizio 1.3
Scrivere un main per simulare la gestione della libreria. In particolare, nel main devono essere letti da input la libreria e un libro L; si deve poi verificare se L è presente nella libreria e in tal caso vendere L (cioè rimuoverlo dalla libreria); se L non è presente deve essere scelto a caso e venduto un altro libro nella stessa fascia d’età di L.

ESERCIZIO 2

Esercizio 2.1

Si deve preparare un sistema per registrare l'andamento degli incontri di una famosa squadra di calcio.
In particolare, il sistema deve essere in grado di memorizzare la formazione in campo e i giocatori in panchina, nonché di modellare i seguenti eventi: sostituzione di un giocatore, espulsione, infortunio.
A tale scopo si implementi una classe GIOCATORE i cui dati siano Ruolo, Età e Numero di Maglia. Si dia inoltre una possibile interfaccia per la classe PARTITA che memorizzi gli elenchi dei giocatori presenti sul campo, di quelli in panchina e l'elenco dei giocatori che hanno segnato almeno un gol nella partita.
Inoltre, si implementino almeno i seguenti metodi della classe PARTITA:

- *bool sostituzione(int g1, int g2)* che tolga dal campo *g1* per inserire *g2*. Si ricordi che sono possibili al più tre sostituzioni, quindi dopo la terza il metodo non fa nulla e restituisce falso.
- *void espulsione(int g1)* che tolga dal campo il giocatore g1.
- *void infortunio(int g1)* che tolga dal campo il giocatore infortunato g1 e verifichi se è possibile effettuare la sostituzione. In caso affermativo, lo suggerisce stampando un messaggio su standard output.

Esercizio 2
Scrivere una funzione che ricevuto un elenco di partite stampi sul video le seguenti informazioni:

- Il giocatore più giovane che ha segnato almeno in una partita.
- La partita in cui ci sono state più espulsioni.
- Se c'è stata almeno una partita in cui non c'è stata nessuna sostituzione

ESERCIZIO 3

Si consideri la seguente definizione della classe *insiemeInt*, che consente di rappresentare *insiemi* di numeri interi.

```
const int N=200;
class InsiemeInt {
private:
       int s[200];
       int dim;
public:
       InsiemeInt();
       InsiemeInt(int s1[], int dim);
}
```

Si implementi il costruttore `InsiemeInt(int s1[], int dim);` e si arricchisca la classe con la definizione e l'implementazione di ALMENO i seguenti operatori:

- `operator []` che ricevuto un intero "i" restituisce l'i-esimo elemento dell'insieme;
- *`insiemeInt&`* `intersezione (`*`insiemeInt&`* `ins2)` che ricevuto un oggetto ins2 di tipo *insiemeInt* restituisce un nuovo oggetto di tipo *insiemeInt* contenente tutti e soli gli elementi presenti contemporaneamente in ins2 e nell'oggetto su cui è stata invocata la funzione. Si supponga che i due insiemi non contengano elementi duplicati.
- `operator >` che confronta due insiemi (ad esempio "a > b") e restituisce TRUE se la somma degli elementi di a è maggiore di quella degli elementi di b, FALSE altrimenti.

ESERCIZIO 4

La nostra cara amica Renata L'Imbranata da qualche mese lavora come dog sitter e, nonostante abbia combinato un bel po' di pasticci, i suoi affari vanno molto bene: non ci sono altri dog sitter nel quartiere in cui vive e perciò tutti si rivolgono a lei. Riceve ogni giorno tantissime richieste e Renata non riesce più a raccapezzarsi, dimentica gli appuntamenti, confonde cani etc. Disperata, Renata ci chiede di fornirle un sistema per la gestione della sua giornata di lavoro.

Esercizio 1
Si implementi una classe APPUNTAMENTO i cui dati rappresentino le informazioni sugli appuntamenti della giornata. In particolare, la classe deve consentire di rappresentare almeno i seguenti dati:

- Il nome del cane
- La razza
- Numero di identificazione
- L'orario a partire dal quale il cane deve essere preso in custodia
- L'orario di riconsegna del cane

Per la classe, oltre ad eventuali costruttori, distruttori e altre funzioni di utilità, si implementino:

- L' Operatore == che restituisca true se due cani hanno lo stesso nome e sono della stessa razza
- L' Operatore >>
- Una funzione che restituisca la durata dell'appuntamento per un dato cane
- L' Operatore > che, dati due cani A e B, consenta di stabilire se la durata dell'appuntamento di A è maggiore, minore o uguale di quella di B.

Esercizio 2
Si implementi una classe AGENDA che contenga l'elenco degli appuntamenti per una data giornata. Per la classe, oltre ad eventuali costruttori, distruttori e altre funzioni di utilità, si implementino:

- L'operatore >>
- Una funzione che restituisca il numero di appuntamenti in una data giornata.
- Una funzione che consenta l'inserimento di un appuntamento nella giornata (Renata non vuol far confusione e perciò non vuole tenere più di un cane per volta. L'inserimento quindi puo' avvenire solo se l'appuntamento non è in concomitanza con altri.
- Una funzione per la cancellazione di un dato appuntamento
- Una funzione che restituisca la razza per la quale ci sono stati più appuntamenti.
- Una funzione che calcoli la durata media di ogni appuntamento.
- Un funzione che controlli se sono presenti nella giornata appuntamenti per cani aventi lo stesso nome e la stessa razza.

Esercizio 3
Si definisca un main in cui si leggano da input un'agenda e un appuntamento A e si verifichi se è possibile inserire A in agenda. Se non è possibile, si cancellino tutti gli appuntamenti in concomitanza con A e si inserisca A.
Si stampi sul video il nome e la razza del cane che deve essere preso in custodia per più tempo e si verifichi inoltre se c'è un cane per cui sono previsti due appuntamenti nella stessa giornata; in caso affermativo stampare sul video il numero identificativo del cane .