Peer-review

Generale

- Nel programma sono stati dati dei nomi poco significativi alle variabili
- `"../include/Bookshelf.h"` inutile, legge lo stesso l'header file
- Il fatto che il programma sia interattivo non consente il testing di tutte le funzioni messe a disposizione da esso. Piuttosto, in questo caso, sarebbe stato meglio creare un "main di test".

Bookshelf

1. Nel seguente costruttore:

```
BookShelf::BookShelf(int a) { // Costruttore che riceve la lunghezza dell'utente
    vector_length = a;
    max_length = 2*vector_length;
    elem = new Book[max_length];
    reserve(minimumL);
}
Copia
```

- Viene impostato il numero di elementi che è `vector_lenght` ad `a`, che è la dimensione specificata dall'utente. Di conseguenza, il numero di elementi occupati non è zero, ma è già `a`, nonostante la libreria sia vuota.
- Cosa fa di preciso `reserve(minimumL)`?
- Manca il check per `size==0`

2. Nel seguente costruttore con l'`initializer_list`

```
BookShelf::BookShelf(std::initializer_list<Book> lst) { // initializer list
    vector_length = lst.size(); //X
    elem = new Book[vector_length];
    std::copy(lst.begin(), lst.end(), elem);
}
Copia
```

- `vector_length = lst.size();` va bene, ma manca 'max_length = lst.size();

3. In `Book& BookShelf::at(int a)`, la classe `Invalid()` avrebbe senso come idea gestire l'errore, però va implementata. Una classe vuota per lanciare un errore secondo noi è inutile.

```
Book BookShelf::at(int a) const { // ritorna elem in posizione inserita dall'utente
    if (a < vector_length && a>=0)
        return elem[a];
    else
        throw Invalid(); // va bene, però...
}
Copia
```

4. In `push_back()`: consiglio: richiamare il metodo `reserve()` invece di riscriverne il codice

```
void BookShelf::push_back(Book b) {
    if (max_length == 0) {
        max_length++;
        elem = new Book[max_length];
    }
    if (vector_length == max_length) {
        max_length = 2*max_length;            //Queste due righe sono implementate
        Book* p = new Book[max_length];       //in reserve() --> codice ripetuto
        std::copy(elem, elem + vector_length, p);
        delete[] elem;
        elem = p;
    }
    elem[vector_length] = b;
    vector_length++;
}
Copia
```

5. Suggerimento: nel file header `Bookshelf.h` avrei dato un nome diverso a `vector_length` e `max_lenght`, perché ambigui e possono essere confusi anche da chi legge il codice.

6. `Book at(int a) const`: non è necessario(?) nella classe del Vector non è presente. *da rivedere*

7. Il metodo `print()` andrebbe fatto con l'overload di `operator<<`

8. Nel 'overload=:

    - Di per se è funzionante, però scrivere `delete[] elem` nella prima riga non è ideale, andrebbe fatto dopo la copia
    - Quel `for` che copia gli elementi di v in `elem` funziona ma sarebbe meglio usare `copy(...)`

```
BookShelf& BookShelf::operator=(const BookShelf& v) {
```

```
    delete[] elem;
    max_length = v.max_length;
    vector_length = v.vector_length;
    elem = new Book[max_length];
    for (int i = 0; i < vector_length; i++)
        elem[i] = v[i];
    return *this;

}
Copia
```

9. Nell'operatore di movimento, stesso problema di modificare `*this`
10. Il distruttore non ha `elem = nullptr`

```
BookShelf::~BookShelf() { // distruttore
    delete[] elem;
    // elem = nullptr <-- MANCANTE
}
Copia
```

Book

- Controllo dell'ISBN solo sulla lunghezza, ma non sui caratteri (ad esempio posso inserire lettere a caso e il programma lo accetterebbe)

Date

- L'overloading di `operator=` non è necessario, in quanto non ritorna un riferimento
- Qual è il senso di fare l'overload di `operator<<` solo per il Month? Lo static_cast è preferibile non usarlo

---

Conclusioni

Dopo aver esaminato il progetto di programmazione in C++, è evidente che, nonostante alcune correzioni da apportare in vari punti, il programma compila ed esegue correttamente, svolgendo le operazioni previste. Alcuni consigli utili basati su questa revisione includono:

- Migliorare la chiarezza dei nomi delle variabili, evitando denominazioni generiche come `a`, `b`, `pippo`, `paperino`, e cercando di evitare nomi simili per variabili che hanno scopi diversi (ad esempio, `vector_length` e `max_length`).

- Nei progetti che richiedono dimostrazioni di corretto funzionamento delle funzioni implementate, è consigliabile creare un main dimostrativo e di test. Tale main, una volta compilato ed eseguito, deve mostrare in modo esplicito le azioni compiute dalle funzioni. Sebbene il main interattivo possa essere piacevole, risulta meno funzionale per chi deve effettuare i test.
- Prestare particolare attenzione agli errori associati alla teoria appresa durante le lezioni, specialmente quelli riscontrati nella classe Bookshelf. Anche se il programma può funzionare in modo apparentemente corretto, errori legati alla teoria possono causare problemi inaspettati.

Questi suggerimenti mirano a migliorare la leggibilità, la funzionalità e la correttezza del vostro progetto e di quelli futuri.