ArrayListBible Algorithm Analysis Worksheet

The following methods are from ***ArrayListBible***. Recall that this class has an ***ArrayList<Verse>*** called ***theVerses***. Assume ***theVerses*** has ***n*** Verses. Give tight bounds for the number of operations required for each. Then give what the ideal implementation would require. Briefly justify. A few have been done for you.

1. Assume that *ref* is the ***m***th verse in *theVerses*. Operations = $\Theta(n)$ Ideal= $\Theta(m)$.

```
public boolean isValid(Reference ref) {
           boolean valid = false;
           if (ref.getBookOfBible() != BookOfBible.Dummy) {
               for (Verse v : theVerses) {
                   if (v.getReference().equals(ref)) {
                       valid = true;
                   }
               }
           }
           return valid;
   }
```

*Since this method loops through the entire ArrayList and does a constant amount of work each time, it takes some constant times **n** operations.*

Ideally the method would return as soon as it determined the verse was valid. This would require Θ(m) operations.

2. Assume that *ref* is the ***m***th verse in *theVerses*. Operations =_____ Ideal =_____

```
public Verse getVerse(Reference ref) {
    for (Verse v : theVerses) {
        if (v.getReference().equals(ref)) {
            return v;
        }
    }
    return null;
}
```

3. Assume that there are ***v*** verses in the given chapter and that the verses for this chapter start at the ***m***th verse in *theVerses*.

Operations =_____ Ideal =_____

```
public int getLastVerseNumber(BookOfBible book, int chapter) {
    int verse = 1;
    while (getVerse(new Reference(book, chapter, verse)) != null) {
        verse++;
    }
    return verse - 1;
}
```

4. Assume that there are ***c*** chapters and a total of ***k*** verses in the entire book and that the verses for this book start at the ***m***th verse in *theVerses*.

Operations = $\Theta(cn)$ Ideal= $\Theta(m+k)$

```
public int getLastChapterNumber(BookOfBible book) {
    int chapter = 1;
    while (isValid(new Reference(book, chapter, 1))) {
        chapter++;
    }
    return chapter - 1;
}
```

This method calls isValid c times, each time requiring $\Theta(n)$ operations. Thus, it takes about $\Theta(cn)$ operations.
Ideally we would just scan until we found the book, requiring $\Theta(m)$ operations, and then scan from there until we found the end of the book, requiring an additional $\Theta(k)$ operations. The total would be $\Theta(m+k)$.

5. Assume the *references* has ***k*** elements and that on average each is at about the middle of *theVerses*.

Operations =_____ Ideal =_____

```
public VerseList getVerses(ArrayList<Reference> references) {
    VerseList results = new VerseList(version, "");
    for (Reference ref : references) {
        if (isValid(ref)) {
            results.add(getVerse(ref));
        } else {
            results.add(null);
        }
    }
    return results;
}
```

6. Assume that *verse1* is the ***m***th verse, there are ***k*** verses between *verse1* and *verse2*, the given book contains ***c*** chapters and that on average each chapter contains ***v*** verses.

Operations = $\Theta(kvm+ccn)$ Ideal= $\Theta(m+k)$

```
ArrayList<Reference> getReferencesInclusive(Reference verse1, Reference verse2) {
    ArrayList<Reference> rList = new ArrayList<Reference>();
    if (verse1.compareTo(verse2) > 0 || !isValid(verse1) || !isValid(verse2)) {
        return rList;
    }
    BookOfBible book = verse1.getBookOfBible();
    int chapter = verse1.getChapter();
    int verse = verse1.getVerse();
    while (!verse2.equals(new Reference(book, chapter, verse))) {
        if (verse == getLastVerseNumber(book, chapter) + 1) {
            chapter++;
            verse = 1;
            if (chapter == getLastChapterNumber(book) + 1) {
                book = BookOfBible.nextBook(book);
                chapter = 1;
            }
        }
        rList.add(new Reference(book, chapter, verse));
        verse++;
    }
    if (verse1.equals(verse2)) {
        rList.add(verse1);
    } else {
        rList.add(verse2);
    }
    return rList;
}
```

This method calls getLastChapterNumber about c times, each time requiring $\Theta(cn)$ time. This takes $\Theta(ccn)$ time total. It also calls getLastVerseNumber for every verse between verse1 and verse2, each time requiring $\Theta(vm)$ time. This takes a total of $\Theta(kvm)$ time. The other operations take constant time for each verse, adding just an additional $\Theta(k)$ time. Thus, the total time required is $\Theta(kvm+cn+k)=\Theta(kvm+ccn)$.

Ideally we would scan until we found verse1 and then scan from there until we found verse2, adding each verse in between to our ArrayList. The first part of the scan would take $\Theta(m)$ operations and the in between part would take $\Theta(k)$ operations for a total of $\Theta(m+k)$.

7. Assume that *verse1* is the ***m***th verse, there are ***k*** verses between *verse1* and *verse2*, the given book contains ***c*** chapters and that on average each chapter contains ***v*** verses.

Operations =_____ Ideal =_____

```
public VerseList getVersesInclusive(Reference verse1, Reference verse2) {
    ArrayList<Reference> refs = getReferencesInclusive(verse1, verse2);
    VerseList verses = getVerses(refs);
    return verses;
}
```

8. Assume that *verse1* is the ***m***th verse, there are ***k*** verses between *verse1* and *verse2*, the given book contains ***c*** chapters and that on average each chapter contains ***v*** verses.

Operations =_____ Ideal =_____

```
ArrayList<Reference> getReferencesExclusive(Reference verse1, Reference verse2) {
    ArrayList<Reference> results = new ArrayList<Reference>();
    if (verse1.compareTo(verse2) > 0) {
        return results;
    }
    int i = 0;
    while(i<theVerses.size()&&!theVerses.get(i).getReference().equals(verse1)) {
        i++;
    }
    while(i<theVerses.size()&&theVerses.get(i).getReference().compareTo(verse2)<0) {
        results.add(theVerses.get(i).getReference());
        i++;
    }
    return results;
}
```

9. Assume that *verse1* is the ***m***th verse, there are ***k*** verses between *verse1* and *verse2*, the given book contains ***c*** chapters and that on average each chapter contains ***v*** verses.

Operations =_____ Ideal =_____

```
public VerseList getVersesExclusive(Reference verse1, Reference verse2) {
    ArrayList<Reference> refs = getReferencesExclusive(verse1, verse2);
    VerseList verses = getVerses(refs);
    return verses;
}
```

10. Assume that verses for the book start at the ***m***th verse, there are ***k*** verses in the book, and that *getReferencesForBook* takes $\Theta(n)$ time.

Operations =_____ Ideal =_____

```
public VerseList getBook(BookOfBible book) {
    return getVerses(getReferencesForBook(book));
}

}
```