Lab 1

Name: Srivenkatesh Nair
Student ID: 378000329
Username: sn6711
ISTE.612

Task 1 (Building the Inverted Index dictionary):

```
Inverted Index:
plot            0       2       4
teen            0       2       3
coupl           0
a               0       1       2       3       4
church          0
parti           0
drink           0
drive           0
accid           0       4
gui             0
di              0
girlfriend      0
continu         0
life            0
nightmar        0
deal            0
watch           0       2       3       4
movi            0       1       2       4
sorta           0
find            0       3
critiqu         0
mind            0
fuck            0
gener           0
touch           0
cool            0       2
idea            0
present         0
bad             0       3       4
packag          0
make            0       2
review          0       1
harder          0
write           0       4
i               0       2       3
```

Task 2 (Query Search):

Testcases:

```
//Queries to be searched
String query1 = new String(original:"wisdom");
String query2 = new String(original:"plot");
String[] query3 = {"strange","thing"};
String[] query4 = {"film","review"};
String[] query5 = {"a","good","start"};
String[] query6 = {"american","thrilling","happy","chase"};
```

```
//Implementing each search strategy
ArrayList<Integer> result1 = inverted.oneWordSearch(query1);
ArrayList<Integer> result2 = inverted.oneWordSearch(query2);
ArrayList<Integer> result3 = inverted.andSearch(query3);
ArrayList<Integer> result4 = inverted.andSearch(query4);
ArrayList<Integer> result5 = inverted.orSearch(query3);
ArrayList<Integer> result6 = inverted.orSearch(query4);
ArrayList<Integer> result7 = inverted.andSearch(query5);
ArrayList<Integer> result8 = inverted.andSearch(query6);
```

```
Terms and their document ids:
wisdom: [2]
plot: [0, 2, 4]
strange: [0, 1]
thing: [0, 1, 2, 3]
film: [0, 2, 3, 4]
review: [0, 1]
a: [0, 1, 2, 3, 4]
good: [0, 1, 3, 4]
start: [0, 1, 4]
american: [0, 2]
thrilling: [0, 3]
happy: [1]
chase: [0, 1, 2]
```

1) Implement a search algorithm that can handle a query with a single keyword:

```
One Word Search -
Document Number: [2] Document names: cv002_17424.txt Words:wisdom

One Word Search -
Document Number: [0, 2, 4] Document names: cv000_29416.txt cv002_17424.txt cv004_12641.txt Words:plot
```

2) Implement a search algorithm that can handle a query with two keywords. Assume that query terms are connected using the AND operator:

```
Two Word (AND) Search -
Document Number: [0, 1] Document names: cv000_29416.txt cv001_19502.txt Words:strange thing

Two Word (AND) Search -
Document Number: [0] Document names: cv000_29416.txt Words:film review
```

3) Implement a search algorithm that can handle a query with two keywords. Assume that query terms are connected using the OR operator:

```
Two Word (OR) Search -
Document Number: [0, 1, 2, 3] Document names: cv000_29416.txt cv001_19502.txt cv002_17424.txt cv003_12683.txt Words:strang
e thing

Two Word (OR) Search -
Document Number: [0, 1, 2, 3, 4] Document names: cv000_29416.txt cv001_19502.txt cv002_17424.txt cv003_12683.txt cv004_126
41.txt Words:film review
```

4) Implement a search algorithm that can handle a query with three or more keywords. Assume that query terms are connected using the AND operator:

```
Multi Word (AND) Search -
Document Number: [0, 1, 4] Document names: cv000_29416.txt cv001_19502.txt cv004_12641.txt Words:a good start

Multi Word (AND) Search -
Words:american thrilling happy chase
No match!
```