**Exercise 7.8 Continued**

1. str.replace(‘ ’,’\n’);
2. str.indexOf(“the”);
3. str.indexOf(“the”, str/length() / 2);
4. //incrementing value

int i = 0;

//first letter

int j = 0;

while(j < (str.length() / 2)){

//last letter

int k = str.index('', i);

if (k == -1){

end = str.length();

//first and last letter substring within str

String word = str.substring(j, k);

//if the string is the

if(word.equals("the")){

i++;

}

//makes the beginning the first letter because -1 is nonexistent

j = k +1;

}

}

**Review Questions 1-8**

1. ! && ||

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **P || !Q** |
| T | T | T |
| F | T | F |
| T | F | T |
| F | F | T |

1. True
2. if(x > min && x < max)
3. if(x > min){

if(x < max){

//do stuff here

}

}

1. for(int i = 1; i <= 10; i++){

for(int j = 1; j <= 10; j++){

System.out.print(“\*”);

}

System.out.println();

}

1. An assertion can be used to check where and if a bug is occurring in the code; for example if a number is too large or doesn’t make sense.
   1. assert (Integer.MAX\_VALUE - a >= b) : "Value of " + a + " + " + b + " is too large to add.";
2. A variant assertion is used to check multiple times throughout the loop, through each change to the variable. An invariant assertion is check typically once and is used to make sure where bugs could happen; similar to that of a variant assertion but just with one check.