1. True or False: To see if variable `height` is equal or greater than variable `width`, we could use:

```java
if (height => width) {
    System.out.println("height => width");
}
```

a) True
b) False because it should be "+=" instead of "+=".

2. True or False: The following code results in multiple lines of output:

```java
System.out.print("Pos ");
System.out.print(" or ");
System.out.print("Neg");
```

a) True
b) False because `print()` will append to the same line. `println()` would have printed a new line character at the end.

3. True or False: The following code results in multiple lines of output:

```java
System.out.print("Pos\n or \nNeg");
```

a) True because `\n` is the new line character.
b) False

4. True or False: The following variable declaration compiles:

```java
int Int;
```

a) True because `Int` is capitalized and is not a keyword.
b) False

5. True or False: The following variable declaration compiles:

```java
int SIXTH_ITEM;
```

a) True because capital letters and underscores are allowed in variable names. Variable names could start with underscores as well.
b) False

6. True or False: The following variable declaration compiles:

```java
int sixth-item;
```

a) True
b) False because dashes/hyphens are not allowed in variable names.

7. True or False: The following variable declaration compiles:

```java
int 6thItem;
```

a) True
b) False because variable names cannot start with a number.
8. What does the following statement print to console?

```
System.out.println(4 >= 5 || !(5 >= 4) || !true != !false && 5 > 4);
```

a) true as shown in the following simplification

```
System.out.println(4 >= 5 || !(true) || true && true);
```

logical negation has higher precedence than relational ops

```
System.out.println(4 >= 5 || !true && true);
```

relational operators has higher precedence than logical AND/OR

```
System.out.println(false || true && true);
```

logical AND has higher precedence than logical OR

```
System.out.println(false || true);
```

logical OR is executed from left to right

```
System.out.println(true);
```

b) false
c) 1
d) 0
e) none of the above

9. What does the following statement print to console?

```
System.out.println(1 + 2 + 3);
```

a) 1 + 2 + 3

b) 123

c) 6 as shown in the following simplification

```
System.out.println(3 + 3);
```

plus gets executed from left the right

```
System.out.println(6);
```

d) 6.0

e) None of the above

10. What does the following statement print to console?

```
System.out.println("" + 1 + 2 + 3);
```

a) 1 + 2 + 3

b) 123 as shown in the following simplification

```
System.out.println("1" + 2 + 3);
```

plus gets executed from left the right

```
String plus an int is concatenation
```

```
System.out.println("12" + 3);
```

```
System.out.println("123");
```

c) 15
d) 6
e) None of the above
11. What does the following statement print to console?

```java
System.out.println(1 + 2 + "3");
```

a) 1 + 2 + 3  
b) 123  
c) 33 as shown in the following simplification

```java
System.out.println(1 + 2 + "3");
    plus gets executed from left the right
System.out.println(3 + "3");
    int plus String is concatenation
System.out.println("33");
```

d) 6  
e) None of the above

12. What does the following statement print to console?

```java
System.out.println(1 + "2" + 3);
```

a) 1 + 2 + 3  
b) 123 as shown in the following simplification

```java
System.out.println(1 + "2" + 3);
    plus gets executed from left the right
System.out.println("12" + "3");
    int plus String is concatenation
System.out.println("123");
String plus int is also concatenation
```

c) 6  
d) None of the above

d) None of the above

13. What does the following statement print to console?

```java
System.out.println(1 + 2 / 3);
```

a) 1 + 2 / 3  
b) 1 as shown in the following simplification

```java
System.out.println(1 + 2 / 3);
    division has higher precedence than addition
    integer division truncates the remainder
System.out.println(1 + 0);
System.out.println(1);
```

c) 1.666[...]  
d) 2  
e) None of the above
14. What does the following statement print to console?

```java
System.out.println((double) 1 + 2 + 3);
```

a) 1 + 2 + 3  
b) 1.023  
c) 123.0  
d) 6.0 as shown in the following simplification

```java
System.out.println((double) 1 + 2 + 3);
```

  - type casting has a higher order of precedence than addition
  ```java
  System.out.println(1.0        + 2 + 3);
  ```
  - adding a double to an int results in a double
  ```java
  System.out.println(3.0            + 3);
  System.out.println(6.0               );
  ```

e) None of the above

15. What does the following statement print to console?

```java
System.out.println("" + 1 + (double) 2 + 3);
```

a) 1 + 2 + 3  
b) 12.03 as shown in the following simplification

```java
System.out.println("" + 1 + (double) 2 + 3);
```

  - type casting has a higher order of precedence than addition
  ```java
  System.out.println("" + 1 + 2.0        + 3);
  ```
  - plus gets executed from left the right
  ```java
  System.out.println("1"    + 2.0        + 3);
  System.out.println("12.0"              + 3);
  System.out.println("12.03"                );
  ```

c) 123.0  
d) 6.0  
e) None of the above

16. What does the following statement print to console?

```java
System.out.println("precaution".substring(3));
```

a) precaution  
b) ecaution  
c) caution because 'p' is index 0, 'r' is index 1, 'e' is index 2, 'c' is index 3. In the substring  
   method, the beginning index is inclusive while the ending index ("precaution".length in  
   this case) is exclusive

d) aution  
e) none of the above
17. What do the following statements print to console?

```java
int a = 3;
int b = 7;
int c = a+++b;
System.out.println("a = " + a + ", b = " + b + ", c = " + ++c);
```

- a) a = 3, b = 8, c = 11
- b) a = 3, b = 8, c = 12
- c) a = 4, b = 7, c = 11
- d) a = 4, b = 7, c = 12
- e) none of the above

The output is calculated as follows:

- `a = 3`
- `b = 7`
- `c = a + b = 3 + 7 = 10`
- `a = a + 1 = 3 + 1 = 4` to apply post-increment
- `++c = c + 1 = 10 + 1 = 11` to increment before accessing/printing
- Finally, printing the values of `a`, `b`, and `c` gives 4, 7, and 11

18. What do the following statements print to console?

```java
if (false)
    System.out.print("not");
System.out.print("with");
System.out.println("standing");
```

- a) notwithstanding
- b) withstanding because only the next statement or set of curly braces is dependent on the if statement.
  Unlike Python or other programming languages, indentation does not affect program execution in Java.
  Because the condition is false, the statement within the if is not executed so "not" will not be printed.
- c) standing
- d)
- e) none of the above
19. What do the following statements print to console?

```java
if (false) {}
    System.out.print("not");
    System.out.print("with");
System.out.println("standing");
```

a) notwithstanding because only the next statement or set of curly braces is dependent on the if statement. There is curly braces that happens to be empty in this case. Therefore, the empty curly braces will not be executed, but all other statements will be. Side note: this is very similar to putting a semi-colon on the if statement as shown below

```java
if (expression);
    // the statement here will be executed
    // regardless of whether expression is true or false.
```

b) withstanding
c) standing
d)
e) none of the above

20. What do the following statements print to console?

```java
if ('0' + 1 == 1) {
    System.out.print("==");
}
System.out.print("!=");
```

a) ==
b) != because '0' + 1 = '1' or 49. Since 49 != 1, the statement in the if block will not execute.
c) ==!=
d) none of the above

21. What do the following statements print to console?

```java
String thing1 = "wombat";
String thing2 = "wombat";
if (thing1.equals(thing2)) {
    System.out.print("==");
} else {
    System.out.print("!=");
}
```

a) == because using the equals() method is the correct way to compare string values.
b) !=
c) ==!=
d) none of the above
22. What do the following statements print to console?

```java
char a = 'o';
switch (a) {
    case 'a': System.out.print("m");
    case 'e': System.out.print("a");
    case 'i': System.out.print("y");
    case 'o': System.out.print("b");
    case 'u': System.out.print("e");
}
System.out.println(" yourself");
```

a) maybe yourself
b) be yourself because the program will start at case 'o' and print out "b" because char a == 'o'. The program will continue to execute the next statement and print out 'e' because there is no break statement. Finally, the program will print out " yourself" regardless.

c) b yourself
d) yourself
e) none of the above

23. What do the following statements print to console?

```java
int i = 17;
while (i > 5) {
    System.out.print("*");
    i -= 3;
}
System.out.println(i);
```

a) *****2
b) *****5
c) ****2
d) ****5
i = 17;
print("*");
i = i - 3 = 17 - 3 = 14;
print("*");
i = i - 3 = 14 - 3 = 11;
print("*");
i = i - 3 = 11 - 3 = 8;
print("*");
i = i - 3 = 8 - 3 = 5, which breaks out of the while loop;
println(i), which is 5;
concatenating all the print statements should results in *****5
e) none of the above
24. What do the following statements print to console?

```java
int i = 5;
doi {
    int j = i + 3;
doi {
        System.out.print(j + " ");
        j = j + 2;
    } while (j < 11);
i = i + 5;
} while (i <= 15);
```

a) 8 10  
b) 8 10 13  
c) 8 10 13 18  

i = 5;  
j = i + 3 = 5 + 3 = 8;  
print(j), which prints 8;  
j = j + 2 = 8 + 2 = 10, which continues the inner do-while loop;  
print(j), which prints 10;  
j = j + 2 = 10 + 2 = 12, which breaks out of the inner do-while loop;  
i = i + 5 = 5 + 5 = 10, which continues the outer do-while loop;  
j = i + 3 = 10 + 3 = 13;  
print(j), which prints 13;  
j = j + 2 = 13 + 2 = 15, which breaks out of the inner do-while loop;  
i = i + 5 = 10 + 5 = 15, which continues the outer do-while loop;  
j = i + 3 = 15 + 3 = 18;  
print(j), which prints 18;  
j = j + 2 = 18 + 2 = 20, which breaks out of the inner do-while loop;  
i = i + 5 = 15 + 5 = 20, which breaks out of the inner do-while loop;  
concatenating all the prints gives "8 10 13 18 ":

d) 8 10 13 18 23  
e) none of the above
25. What do the following statements print to console?

```java
int sum = 0;
for (int i = 0; i < 3; ++i) {
    sum += i;
}
System.out.println(sum);
```

a) 1
b) 2
c) 3
   sum = 0;
   i = 0;
   sum = sum + i = 0 + 0 = 0;
   i = i + 1 = 0 + 1 = 1;
   sum = sum + i = 0 + 1 = 1;
   i = i + 1 = 1 + 1 = 2;
   sum = sum + i = 1 + 2 = 3;
   i = i + 1 = 2 + 1 = 3, which breaks out of the while loop;
   println(sum), which is 3;
d) 6

e) none of the above
26. What do the following statements print to console?

```java
for (int i = 0; i < 8; ++i) {
    if (i % 3 == 1 || i < 2) {
        continue;
    }
    System.out.print(i + " ");
}
```

a) 0 1 4 7  
b) 2 3 5 6  
   i = 0;
   i < 2, so continue;
   i = i + 1 = 0 + 1 = 1;
   i < 2, so continue;
   i = i + 1 = 1 + 1 = 2;
   both i%3==1 and i<2 is false, so don’t continue;
   print(i), which is 2;
   i = i + 1 = 2 + 1 = 3;
   both i%3==1 and i<2 is false, so don’t continue;
   print(i), which is 3;
   i = i + 1 = 3 + 1 = 4;
   i%3==1, so continue;
   i = i + 1 = 4 + 1 = 5;
   both i%3==1 and i<2 is false, so don’t continue;
   print(i), which is 5;
   i = i + 1 = 5 + 1 = 6;
   both i%3==1 and i<2 is false, so don’t continue;
   print(i), which is 6;
   i = i + 1 = 6 + 1 = 7;
   i%3==1, so continue;
   i = i + 1 = 7 + 1 = 8, which breaks out of the for loop;
   concatenating all the prints gives "2 3 5 6"

c) 2 3 5 6 7  
d) 0 1 2 3 4 5 6 7  
e) none of the above
27. What do the following statements print to console?

```java
for (int i = 0; i < 4; ++i) {
    if (i == 2) {
        if (true) {
            break;
        }
    }
    System.out.print(i + " ");
}
```

a) 0 1
   i = 0;
   i != 2, so don’t execute the break;
   print(i), which is 0;
   i = i + 1 = 0 + 1 = 1;
   i != 2, so don’t execute the break;
   print(i), which is 1;
   i = i + 1 = 1 + 1 = 2;
   i == 2 and true, so do execute the break, which breaks out of the entire for loop;
   concatenating all the prints gives "0 1"

b) 0 1 3

c) 0 1 2 3

d) 0 1 3 4

e) none of the above
28. What do the following statements print to console?

```java
int a = 2;
int b = 0;
for (int i = 1; i <= 3; ++i) {
    a = i;
    if (i % 2 == 0) {
        a = i * -1;
    }
    b = b + a;
}
System.out.println(b);
```

a) 1  
b) -1  
c) 2  
   a = 2;  
   b = 0;  
   i = 1;  
   a = i = 1;  
   i%2 != 0, so don’t negate a;  
   b = b + a = 0 + 1 = 1;  
   i = i + 1 = 1 + 1 = 2;  
   a = i = 2;  
   i%2 == 0, so a = i * -1 = 2 * -1 = -2;  
   b = b + a = 1 + -2 = -1;  
   i = i + 1 = 2 + 1 = 3;  
   a = i = 3;  
   i%2 != 0, so don’t negate a;  
   b = b + a = -1 + 3 = 2;  
   i = i + 1 = 3 + 1 = 4, which breaks the for loop;  
   println(b), which is 2;  

d) -2  
e) none of the above
29. What do the following statements print to console?

```java
for (int i = 3; i < 5; ++i) {
    for (int j = 6; j >= 4; j -= 2) {
        System.out.print(i * j + " ");
    }
    System.out.println(" ");
}
```

a) 18 12
   24 16
   i = 3;
   j = 6;
   print(i*j), which is 3*6 = 18;
   j = j - 2 = 6 - 2 = 4;
   print(i*j), which is 3*4 = 12;
   j = j - 2 = 4 - 2 = 2, which break out of the inner for loop;
   println(" ");
   i = i + 1 = 3 + 1 = 4;
   j = 6;
   print(i*j), which is 4*6 = 24;
   j = j - 2 = 6 - 2 = 4;
   print(i*j), which is 4*4 = 16;
   j = j - 2 = 4 - 2 = 2, which break out of the inner for loop;
   println(" ");
   i = i + 1 = 4 + 1 = 5, which breaks out of the outer for loop;
   concatenating all the prints gives "18 12 \n24 16 \n"

b) 18 24
   12 16
c) 18 15 12
   24 20 16
d) 18 15 12
   24 20 16
   30 25 20
e) none of the above
30. What do the following statements print to console?

```java
for (int i = 1; i < 5; ++i) {
    for (int j = 1; j < i; ++j) {
        if (i % j != 0) {
            System.out.print(i + " ");
        }
    }
}
```

a) 3 4

i = 1;
j = 1, which breaks out of the inner for loop;
i = i + 1 = 1 + 1 = 2;
j = 1;
i % j == 0, so don’t execute the print statement;
j = j + 1 = 1 + 1 = 2, which breaks out of the inner for loop;
i = i + 1 = 2 + 1 = 3;
j = 1;
i % j == 0, so don’t execute the print statement;
j = j + 1 = 1 + 1 = 2;
i % j != 0, so do execute the print statement;
print(i), which is 3;
j = j + 1 = 2 + 1 = 3, which breaks out of the inner for loop;
i = i + 1 = 3 + 1 = 4;
j = 1;
i % j == 0, so don’t execute the print statement;
j = j + 1 = 1 + 1 = 2;
i % j == 0, so don’t execute the print statement;
j = j + 1 = 2 + 1 = 3;
i % j != 0, so do execute the print statement;
print(i), which is 4;
j = j + 1 = 3 + 1 = 4, which breaks out of the inner for loop;
i = i + 1 = 4 + 1 = 5, which breaks out of the outer for loop;
concatenating all the prints gives "3 4 "

b) 2 3 4
c) 1 2 3 4
d) 2 3 4 4
e) none of the above
31. What do the following statements print to console?

```java
int[] intArray = {1, 2, 3, 7, 8, 6, 7, 8, 4, 5};
for (int index = 2; index < intArray.length; index += 3) {
    System.out.print(intArray[index] + " ");
}
```

a) 2 5 8
b) 2 8 8
c) 3 6 4
index = 2;
    print(intArray[index]), which is intArray[2], which is 3;
    index = index + 3 = 2 + 3 = 5;
    print(intArray[index]), which is intArray[5], which is 6;
    index = index + 3 = 5 + 3 = 8;
    print(intArray[index]), which is intArray[8], which is 4;
    index = index + 3 = 8 + 3 = 11, which breaks the for loop;
    concatenating all the prints gives "3 6 4"

d) 3 6 9
e) none of the above

32. What do the following statements print to console?

```java
int[] intArray = {3, 4, 6, 7, 8, 9};
for (int index = 0;
     intArray[index] % 3 == 0 || intArray[index] % 4 == 0;
     ++index) {
    System.out.print(index + " ");
}
```

a) 0 1 2
index = 0;
    intArray[index] % 3 == 0 because intArray[0] is 3 and 3%3==0, so continue for loop
    print(index), which is 0;
    index = index + 1 = 0 + 1 = 1;
    intArray[index] % 4 == 0 because intArray[1] is 4 and 4%4==0, so continue for loop
    print(index), which is 1;
    index = index + 1 = 1 + 1 = 2;
    intArray[index] % 3 == 0 because intArray[2] is 6 and 6%3==0, so continue for loop
    print(index), which is 2;
    index = index + 1 = 2 + 1 = 3;
    break out of for loop because intArray[3] is 7, which is neither divisible by 3 nor 4
    concatenating all the prints gives "0 1 2"

b) 1 2 3
c) 0 1 2 4 5
d) 3 4 6 8 9
e) none of the above
33. What do the following statements print to console?

```java
int[] intArray = {3, 4, 6, 8};
for (int index = 0; index < intArray.length; ++index) {
    intArray[index] = intArray[(index + 1) % intArray.length];
}
for (int index = 0; index < intArray.length; ++index) {
    System.out.print(intArray[index] + " ");
}
```

a) 4 6 8 3
b) 4 6 8 4
index = 0;
    intArray[0] = intArray[1] because index is 0, so intArray is now {4, 4, 6, 8};
    index = index + 1 = 0 + 1 = 1;
    intArray[1] = intArray[2] because index is 1, so intArray is now {4, 6, 6, 8};
    index = index + 1 = 1 + 1 = 2;
    intArray[2] = intArray[3] because index is 2, so intArray is now {4, 6, 8, 8};
    index = index + 1 = 2 + 1 = 3;
    intArray[3] = intArray[0] because index is 3, so intArray is now {4, 6, 8, 4};
    index = index + 1 = 3 + 1 = 4, which breaks out of the first for loop;
    the second for loop prints the arrays, which prints("4 6 8 4 ");

c) 8 3 4 6
d) 8 3 4 8
e) none of the above

34. What values of x and y do not result in short circuit evaluation, such that all operands are evaluated?

```
(x >= 3) && (y != 5) && (z > 30)
```

a) x = 2, y = 4
b) x = 2, y = 5
c) x = 3, y = 4
    for logical AND, short circuit evaluations cannot occur if the earlier conditions are true
    therefore, want x >= 3 && y != 5
    x = 3 and y = 4 satisfies x >= 3 && y != 5
d) x = 3, y = 5
e) none of the above
35. Consider the following code:

```java
for (int i = start; i <= end; ++i) {
    for (int j = first; j < last; ++j) {
        System.out.print(i * j + " ");
    }
    System.out.println();
}
```

The output of the code above was:

```
12 16 20 
15 20 25 
18 24 30 
21 28 35
```

What are the values of variables `start`, `end`, `first`, and `last`?

a) `start = 3, end = 7, first = 4, last = 6`
b) `start = 4, end = 7, first = 3, last = 6`
c) `start = 7, end = 3, first = 6, last = 4`
d) `start = 7, end = 4, first = 6, last = 3`
e) none of the above

36. Given string input, what do the following statements print to console?

```java
java.util.Scanner scnr = new java.util.Scanner(System.in);
String userInput = scnr.next();
String newWord = "";
for (int i = userInput.length() - 1; i >= 0; i -= 2) {
    char c = userInput.charAt(userInput.length() - i - 1);
    newWord = newWord + c;
}
System.out.print(newWord);
```

a) Every even character of userInput
b) Every even character of userInput in reverse order
c) Every odd character of userInput
d) Every odd character of userInput in reverse order
e) None of the above
37. Given positive integer input, what do the following statements print to console?

```java
java.util.Scanner scnr = new java.util.Scanner(System.in);
int userInput = scnr.nextInt();
int a = 0;
int b = 1;
for (int i = 0; i < userInput; ++i) {
    System.out.print(a + " ");
    int c = a;
    a = b;
    b += c;
}
```

a) The first userInput numbers raised to the power of 2 (0, 4, 16, 25, ...)  
b) 2 raised to the first userInput numbers (0, 1, 2, 4, 8, 16, ...)  
c) The first userInput Fibonacci numbers (0, 1, 1, 2, 3, 5, ...)  
d) The factorial of the first userInput numbers (1, 1, 2, 6, 24, 120, ...)  
e) None of the above

38. Given positive integer input, what do the following statements print to console?

```java
java.util.Scanner scnr = new java.util.Scanner(System.in);
int userInput = scnr.nextInt();
for (int i = 2; i <= userInput; ++i) {
    boolean flag = true;
    for (int j = 2; j <= i / 2; ++j) {
        if (i % j == 0) {
            flag = false;
            break;
        }
    }
    if (flag) {
        System.out.print(i + " ");
    }
}
```

a) All even numbers up to userInput/2  
b) All factors of userInput  
c) All prime numbers up to userInput  
d) All composite numbers up to userInput  
e) None of the above
39. Given positive integer inputs, what do the following statements print to console?

```java
java.util.Scanner scnr = new java.util.Scanner(System.in);
int userInput1 = scnr.nextInt();
int userInput2 = scnr.nextInt();
int output = 1;
for (int i = 0; i < userInput1; ++i) {
    output = output * userInput2;
}
System.out.println(output);
```

a) userInput1 * userInput2 (userInput1 times userInput2)
b) userInput1 ^ userInput2 (userInput1 to the power of userInput2)
c) userInput2 ^ userInput1 (userInput2 to the power of userInput1)
d) userInput1! * userInput2 ((factorial of userInput1) times userInput2)
e) None of the above

40. Which of the following styles do you think makes the best use of whitespace?

a)
```java
int ellipseMajorRadius = 0; // major radius of ellipse, a in cm
int ellipseMinorRadius = 0; // minor radius of ellipse, b in cm
double ellipseArea = 0.0; // area of ellipse = pi*a*b in cm^2
double ellipseFoci = 0.0; // foci of ellipse, f = sqrt(a^2+b^2) in cm
double ellipseEccentricity = 0.0; // eccentricity of ellipse, e = f/a
double ellipseAdditionalConceptAngularEccentricity = 0.0; // angular eccentricity of ellipse
boolean isCircle = false; // boolean to check whether a == b
```

b)
```java
int ellipseMajorRadius = 0; // major radius of ellipse, a in cm
int ellipseMinorRadius = 0; // minor radius of ellipse, b in cm
double ellipseArea = 0.0; // area of ellipse = pi*a*b in cm^2
double ellipseFoci = 0.0; // foci of ellipse, f = sqrt(a^2+b^2) in cm
double ellipseEccentricity = 0.0; // eccentricity of ellipse, e = f/a
double ellipseAdditionalConceptAngularEccentricity = 0.0; // angular eccentricity of ellipse
boolean isCircle = false; // boolean to check whether a == b
```

c) No wrong answers for this question because this question does ask what do you think. I personally think that choice c looks the best because of its vertical alignment. Also, not shifting all comments across because you there’s one/few long variable name(s) or long statement(s) looks nicer in my opinion.
```java
int ellipseMajorRadius = 0; // major radius of ellipse, a in cm
int ellipseMinorRadius = 0; // minor radius of ellipse, b in cm
double ellipseArea = 0.0; // area of ellipse = pi*a*b in cm^2
double ellipseFoci = 0.0; // foci of ellipse, f = sqrt(a^2+b^2) in cm
double ellipseEccentricity = 0.0; // eccentricity of ellipse, e = f/a
double ellipseAdditionalConceptAngularEccentricity = 0.0; // angular eccentricity of ellipse
boolean isCircle = false; // boolean to check whether a == b
```

d)
```java
int ellipseMajorRadius = 0; // major radius of ellipse, a in cm
int ellipseMinorRadius = 0; // minor radius of ellipse, b in cm
double ellipseArea = 0.0; // area of ellipse = pi*a*b in cm^2
double ellipseFoci = 0.0; // foci of ellipse, f = sqrt(a^2+b^2) in cm
double ellipseEccentricity = 0.0; // eccentricity of ellipse, e = f/a
double ellipseAdditionalConceptAngularEccentricity = 0.0; // angular eccentricity of ellipse, alpha
boolean isCircle = false; // boolean to check whether a == b
```