**LAB EXERCISE 7 – ANSWER Tuğberk GÖÇ - 115200084**

**1)**

**IN JAVA :**

**int** a = 0, b = 0, c = 0;

// Before printing, we have to initialize them to run print function in JAVA.

System.***out***.println(a); // 0

System.***out***.println(b); // 0

System.***out***.println(c); // 0

// We can see that outputs are going to be 0.

a = 94;

b = 94;

c = 94;

System.***out***.println(a); // 94

System.***out***.println(b); // 94

System.***out***.println(c); // 94

// We can see that outputs are going to be 94 because before printing, we assign them to 94.

// Also we can not redefine them as int because they are already defined.

**IN PYTHON:**

a = b = c = 0

# We have to define them, before printing

# As we can see values are going to assign 0.

print(a) # 0

print(b) # 0

print(c) # 0

# We can see that all of them are going to print 0.

a = b = c = 94

print(a)

print(b)

print(c)

# We can see that all of them are going to print 94 because we assign them as 94 on line 12.

**IN C++:**

int a = 0;

int b = 0;

int c = 0;

// Before printing, we have to initialize(typecasting) them to run print function in C++.

std::cout << "\n a = " << a << " b = " << b << " c = " << c;

// We can see that outputs are going to be 0.

a = 94;

b = 94;

c = 94;

std::cout << "\n a = " << a << " b = " << b << " c = " << c;

// We can see that outputs are going to be 94 because before printing, we assign them to 94.

// Also we can not redefine them as int because they are already defined.

// C++ is more closer to Java than Python.

**2)**

**IN C++:**

In C++, if you don’t assign value and just create it, you are going to see such numbers. Where these numbers come from. Read below.

You read uninitialized memory. This is undefined behavior. That means the C++ language rules give literally no guarantee what happens when you run this code, at all. You may see the values you observed, nothing at all, crash your PC or set your house on fire. All equally legal outcomes of running this code.

int c;

int b;

if ( c = b)

std::cout << "\n IF true c= " << c << " b = " << b;

// IF true c= 32764 b =32764

else

std::cout << "\n IF else c= " << c << " b = " << b;

if ( c == b)

std::cout << "\n IF true c= " << c << " b = " << b;

// IF true c= 32764 b =32764

else

std::cout << "\n IF else c= " << c << " b = " << b;

**IN JAVA:**

**int** b = 0;

**int** c = 0;

// Before running in JAVA, we have to assign to value because JAVA do not ability to assign random memory place like C++. Thus, I assigned them in the beginning.

**if** ( c == b)

System.***out***.println("If true c = " + c + " b = " + b);

// If true c = 0 b = 0

**else**

System.***out***.println("If else c = " + c + " b = " + b);

**if** ( c == b)

System.***out***.println("If true c = " + c + " b = " + b);

// If true c = 0 b = 0

**else**

System.***out***.println("If else c = " + c + " b = " + b);

**3)**

**IN C++:**

**PART 01:**

int main() {

int a; // we do not assign to the value. We just do typecasting

int b; // we do not assign to the value. We just do typecasting

int c; // we do not assign to the value. We just do typecasting

int d; // we do not assign to the value. We just do typecasting

a = b + (c = d / b) - 1;

std::cout << "a = " << a; // a = 32763

// We can see 32763 for a’s value because IN C++, values are not defined in beginning, they are going to assign end of the memory. (Last int value)

return 0;

}

**PART 02:**

int main() {

int t;

int y;

int z;

t=(y+5)\*z-1; // x=y+5;

std::cout << "t =" << t; // t = 1223730239

// Again, we do not have to do valuecasting. We can just do typecasting at the beginning to get values end of the memory for C++.

// It can work like that but not properly.

return 0;

}

**IN JAVA:**

**PART 01:**

**public** **static** **void** main(String[] args) {

**int** a = 1;

**int** b = 2;

**int** c;

**int** d = 4;

// We do typecasting and valuecasting. Because, we use JAVA, it is required to do valuecasting as well.

// But, for c, we do not need to do valuecasting. It's assigned by code automatically.

a = b + (c = d / b) - 1;

System.***out***.println(a); // 3

// As you can see, a is going to be 3 because we have done all the work in java such as typecasting and valuecasting.

}

**PART 02:**

**public** **static** **void** main(String[] args) {

**int** t; // we do not need to do valuecasting, not typecasting. Because, we will assign below.

**int** y = 2;

**int** z = 3;

t = (y + 5) \* z - 1; // x=y+5;

System.***out***.println(t); // 20

// As you can see, we are going to see 20 for t.

}

**4)**

**IN C++:**

#include <iostream>

int main() {

int t = 59; int a = 54; int c = 9; // We shoul do typecasting with given these assignees.

std::cout << "\n before if t = " <<t<< " a = " <<a<< " c = " <<c;

// OUTPUT: before if t = 59 a = 54 c = 9

// These values are defined above, they are not changed by anything.

if ((t > a) || (c++/3)) // t > a : true , therefore c is not incremented by 1.

std::cout << "\n in if t = " <<t<< " a = " <<a<< " c = " <<c;

// OUTPUT: in if t = 59 a = 54 c = 9

// Thus, c is not going to change. It protects its own value as 9.

std::cout << "\n after if t = " <<t<< " a = " <<a<< " c = " <<c;

// OUTPUT: after if t = 59 a = 54 c = 9

// Nothing changed. Before coming here. All values are same.

t=50;

std::cout << "\n before if t = " <<t<< " a = " <<a<< " c = " <<c;

// OUTPUT: before if t = 50 a = 54 c = 9

// We just changed t as 50. Therefore, we can see t as 50.

if ((t > a) || (c++/3)) // Not second expression is true (c++/3) : Therefore we increment c by 1.

std::cout << "\n in if t = " <<t<< " a = " <<a<< " c = " <<c;

// OUTPUT: in if t = 50 a = 54 c = 10

// C is changed as 10. Because, we increment by 1 in if statement.

std::cout << "\n after if t = " <<t<< " a = " <<a<< " c = " <<c;

// OUTPUT: after if t = 50 a = 54 c = 10

// All values that are changed in if or main, They all are assigned at global scope. Therefore we can see them changed.

return 0;

}

**IN JAVA:**

**public** **static** **void** main(String[] args) {

**int** t=59; **int** a=54; **int** c=9; // We have to do typecasting after taking these given values.

System.***out***.println("\n before if t = " + t + " a = " + a + " c = " + c);

// before if t = 59 a = 54 c = 9

**if** ((t > a) || (c++%3 == 0)) // JAVA version of second expression : (c++%3 == 0)

System.***out***.println("\n in if t = " + t + " a = " + a + " c = " + c);

// in if t = 59 a = 54 c = 9

// c is not changed because second expression is false.

System.***out***.println("\n after if t = " + t + " a = " + a + " c = " + c);

// after if t = 59 a = 54 c = 9

t=50;

System.***out***.println("\n before if t = " + t + " a = " + a + " c = " + c);

// before if t = 50 a = 54 c = 9

// Just t changed because we assign it to 50. (In Line 15)

**if** ((t > a) || (c++%3 == 0))

System.***out***.println("\n in if t = " + t + " a = " + a + " c = " + c);

// in if t = 50 a = 54 c = 10

// Second expression is true therefore c is incremented by 1.

// We can see c's value as 10. Not 9.

System.***out***.println("\n after if t = " + t + " a = " + a + " c = " + c);

// after if t = 50 a = 54 c = 10

// These values are all changed in global scope.

// Thus, we can see these values as last version.

}

/\*\*

\* If I compare this with C++, They are so similar about running process.

\* There is only one difference that how to write (c++%3 == 0) modulo expression.

\*/