Question 1 (3*3 =9 marks)

Consider the following code

```
1     final int INDEX=10;
2     String name="Tim";
3     int salary;
4     System.out.println(name);
5     INDEX=22;
6     System.out.println("Name:"+Name+"   Salary:"+salary);
7     salary*=1.5;
```

1. A constant is a value that cannot be changed once it has been assigned. In Java, the keyword "final" is used to declare a constant. In line 1, the statement "final int INDEX=10;" declares that the constant name is INDEX, data type is int and data value has been initialised as 10. However, in line 5, the statement "INDEX=22;" tries to reassign the value 22 to the constant INDEX. As constants cannot be changed or reassigned, the statement in line 5 will result in a syntax error.

2. In line 2, the statement "String name="Tim";" declares a variable called "name", which means that we label the memory location as "name". This location stores data with string data type, and the value "Tim" is stored at this memory location. However, in line 6, flowing "Name: ", there is a typing error, which writes "Name" instead of "name". The purpose of the statement "System.out.println ("Name: "+name+" Salary: "+salary);" is to display the value ("Tim") that has been stored in the memory location called "name". If we write "Name" instead of "name", Java will not be able to find a memory location called "Name" and an error will occur.

3. There are two errors related to salary.

First, in line 3, the statement "int salary;" declares a variable named "salary" without being initialised. This means that there is no initial value stored in the memory location labelled as "salary". Because Java executes the codes from top to bottom, when Java tries to execute statement in line 6 (which is to display the value stored in the memory location called "salary"), Java cannot find any value stored in the memory location called "salary". In short, a variable has to be initialised before it is used. Otherwise, a syntax error occurs.

Second, in line 7, "*=" is a multiplication compound assignment operator, and "salary*=1.5;" means "salary=salary*1.5;" and we assign the value from right side operand (salary*1.5) to the left side operand (salary). Again, because we did not initialised the variable salary, this assignment statement cannot be executed successfully. Although the statement "salary*=1.5;" runs successfully if we initialise the value of salary (for example 12), we cannot get our desired output because it is written below "System.out.println( )". As Java executes the codes from top to bottom,

to get the desired outcome, we should write "salary*=1.5;" before we write the statement "System.out.println ("Name: "+name+" Salary: "+salary);".

The following is the code that has been corrected and runs successfully:

```
final int INDEX=10;
String name="Tim";
int salary;
salary=12;
salary*=1.5;
System.out.println(name);
System.out.println("Name: "+name+" Salary: "+salary);
```

Question 2 (2+3=5 marks)

Consider the following code:

```
1    int a=15;
2    int b=4;
3    double result=a/b;
4    System.out.println(result);
```

a) The value of **result** is 3.0 instead of 3.75. Why?

b) Suggest three different solutions to make result=3.75

Q2: your 1st and 3rd are essentially the same

a) In Java, dividing an integer by an integer, the result will be an integer.

To clarify the code, it can be written as:

int a = 15;

int b = 4;

int c = a/b;

double result = c;

System.out.println (result);

When doing the calculation "int c=a/b;", we get 15/4=3.75. Because the data type of variable c is int, only the integer part of 3.75 is kept. Therefore, int c=3.

The statement "double result = c;" can be seen as a conversion from int to double. Because int (lower data type) has smaller memory size than double (higher data type), int can be automatically converted to double. Therefore, the value of result is 3.0.

b)

solution 1. we can declare the variables a and b as double:

double a = 15;

double b =4;

double result=a/b;

System.out.println (result);

solution 2. we can cast 15 and 4 into double:

double a = (double)15;

double b=(double)4;

double result = a/b;

System.out.println (result);

solution 3. we can use float instead of int:

float a = 15F;

float b = 4F;

float result = a/b;

System.out.println (result);

```
public class Test {
    public static void main(String[] args) {
        double a =15;
        double b=4;
        double result=a/b;
        System.out.println(result);
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        double a =15;
        int b=4;
        double result=a/b;
        System.out.println(result);
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        int a =15;
        int b=4;
        double result=1.0*a/b;
        System.out.println(result);
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        double a =15.0;
        double b=4.0;
        double result=a/b;
        System.out.println(result);
    }
}
```

Question 3 (12 marks)

Briefly explain the output of each statement?

```
int aValue = 15;
System.out.println(aValue=21);
System.out.println(aValue!=21);
System.out.println(aValue%7==0);
System.out.println(aValue=aValue-4);
System.out.println(aValue/5);
System.out.println(aValue+"3");
```

The output is: 21, false, true, 17, 3, 173.

1. int aValue=15;

   System.out.println(aValue=21);

  This code can be seen as a combination of the following code:

int aValue=15;

aValue=21;

System.out.println (aValue);

It means that we declares a variable called aValue and we assign an initial value 15 to it. As "=" is a simple assignment operator which assigns values from right side operands to left side operand, "aValue=21;" means that we assign a new value 21 to the variable aValue and now only the new value 21 is stored in the memory location called "aValue". Therefore, when we use println( ) method to display the value stored in aValue, the result is 21.

2. "!=" is a relational operator (not equal to) which checks if the values of two operands are equal or not. If the two values are not equal, the condition becomes true. As mentioned above, the value of aValue has been changed to 21, the statement "aValue!=21" can be understood as "21 does not equal to 21", which is false. Therefore, the result is false.

3."%" is an arithmetical operator which divides the left-hand operand by the right-hand operand and returns the remainder. As mentioned above, the value of aValue is still 21. "aValue%7" means that we divide 21 by 7 and return the remainder, which is 0. "==" is a relational operator (equal to) which checks if the values of two operands are equal or not, if they are equal, then the condition becomes true. Thus, the statement "aValue%7==0" can be understood as checking if "0 equals to 0". Obviously, the statement is correct. Therefore, the result is true.

4. The statement System.out.println(aValue=aValue-4); can be seen as the combination of the following code:

aValue=aValue-4;

System.out.println(aValue);

As mentioned above, the value stored in aValue is 21. "=" is a simple assignment operator which assigns values from right side operands to left side operand. Thus, we calculate the right side operands "aValue-4" and we get 21-4=17. Then the value 17 is assigned to the left side operand (aValue). Now the the value of aValue becomes 17. Therefore, when we executing the code, we will get the new value of aValue, which is 17.

5. The value of the variable aValue is 17. aValue/5=17/5=3.4. In Java, the default data type of integer literals is int, so the data type of 5 is int. As the variable aValue is declared with data type int, aValue/5 can be seen as int divided by int and the result should also be int. Therefore, the result is the integer part of 3.4, which is 3. This is why the output of "System.out.println(aValue/5);" is 3.

6. The data type of aValue is int and the value is 17. Because 3 is written in double quotes, "3" is not an integer. Instead, it is a string. Here, "+" is not an arithmetic operator (addition) because we cannot add a value of data type string and a value of data type int. "+" only means that we concatenate the value of aValue (17) and the string ("3") when displaying these two values. Therefore, the output of the statement "System.out.println (aValue+"3");" is 173.