Final Exam

CSE 1020 3.0

Section M, Winter 2010

Family Name: _____

Given Name(s): _____

Student Number: |__ |__ |__ |__ |__ |__ |__ |__ |__ |

Guidelines and Instructions:

1. This is a 90-minute exam. You can use the textbook, but no electronic aids such as calculators, cellphones etc.
2. Answer questions in the space provided. If you need more space, use the back of the page. Clearly indicate that your answer continues on the back of the page.
3. Write legibly. Unreadable answers will not be marked.
4. Leave your ID on the desk. A sign-up sheet will be distributed during the test. By signing it, you acknowledge that you are registered in the course and you are the owner of the associated ID.
5. Keep your eyes on your own work. At the discretion of the invigilators, students may be asked to move.

Question	Out of	Mark
Q1	34	
Q2	36	
Q3	30	
Total	100	
Letter grade		

Q1. [34 marks] Suppose that you are asked to test an app that accepts as input a numerical range and prints out all the numbers in that range separated by spaces. An example run for such an app would be:

```
Enter a numerical range: 1-9
1 2 3 4 5 6 7 8 9
```

If the bounds of the range are given in descending order, no output should be produced, as in:

```
Enter a numerical range: 13-9
```

The implementation you are testing is shown below. It compiles with no errors. Pages 225-228 in the textbook contain descriptions of the methods used in this app.

```
import java.util.Scanner;
import java.io.PrintStream;

public class Q1
{
  public static void main(String[] args)
  {
    Scanner input = new Scanner(System.in);
    PrintStream output = System.out;

    output.print("Enter a numerical range: ");
    String str = input.nextLine();
    int dash = str.indexOf("-");
    String left = str.substring(0, dash);
    String right = str.substring(dash + 1);
    int begin = Integer.parseInt(left);
    int end = Integer.parseInt(right);
    for (int i = begin; i < end; i++)
    {
      output.print(i + " ");
    }
    output.println(end);
  }
}
```

In the space below and on the next page, describe a set of test cases that you would develop for this app. For each test case, indicate whether it will pass, reveal a logic error, or produce a run-time error. For every test case that does not pass, indicate how you would fix the problem. You do not have to provide exact code, simply a description of the changes you would make.

There is a bug: Input 13-9 produces 13 instead of nothing. Also, if there is no dash or non-numbers, there will be exceptions similar to the example in the textbook. All other test cases should pass.

4 marks: Upper bound > Lower bound. Works ok.

4 marks: Upper bound = Lower bound. Works ok.

6 marks: Upper bound < Lower bound. Produces output instead of nothing. Add if statement.

6 marks: Negative numbers. NumberFormat Exception if the first number is negative, logic error similar to Upper bound < Lower bound if only the second number is negative.

4 marks: Overflow numbers. NumberFormatException

5 marks: No dash. IndexOutOfBoundsException

5 marks: Non-integers around the dash. NumberFormatException

Q2. [36 marks] Consider the API for the three classes `Animal`, `Cat`, and `Dog` below.

public abstract class Animal

This class encapsulates an animal.

Field Detail

public String name

The name of this animal

Method Detail

public abstract void printName()

Prints out information about this animal.

public class Cat extends Animal

This class encapsulates a cat.

Constructor Detail

public Cat(String s)

The name field is assigned the value of s

Parameters:

s - the name of this cat

Method Detail

public void printName()

Always prints out the string "Cat", followed by a space, followed by the value of the name field, followed by a newline.

public void meow()

Always prints out the string "Meow!" followed by a newline.

public class Dog extends Animal

This class encapsulates a dog.

Constructor Detail

public Dog(String s)

The name field is assigned the value of s

Parameters:

s - the name of this dog

Method Detail

public void printName()

Always prints out the string "Dog", followed by a space, followed by the value of the name field, followed by a newline.

public void bark()

Always prints out the string "Arf!" followed by a newline.

Part 1: For each one of the code segments in this and the next page, check the box next to the expected outcome. Follow the instructions given after the selected answer. You can assume all needed classes have been imported. *No marks will be awarded without error explanation or correct output as the case may be.* Each code segment is worth 4 marks.

(a)
```
Animal a1 = new Dog("Moby");
a1.printName();
```

- ☐ Compiles and runs (write output in space below).
- ☐ Compile-time error (explain source of error).
- ☐ Run-time error (explain source of error).

Compiles and runs. Output is:

```
Dog Moby
```

(b)
```
Animal a2 = new Dog("Moby");
a2.bark();
a2.printName();
```

- ☐ Compiles and runs (write output in space below).
- ☐ Compile-time error (explain source of error).
- ☐ Run-time error (explain source of error).

Compile-time error. Method `bark()` is not available for references of type `Animal`.

(c)
```
Dog d3 = new Cat("Astor");
d3.meow();
d3.printName();
```

- ☐ Compiles and runs (write output in space below).
- ☐ Compile-time error (explain source of error).
- ☐ Run-time error (explain source of error).

Compile-time error. Incompatible types in line 1.

(d)
```
Animal a4 = new Animal("Moby");
a4.printName();
```

- ☐ Compiles and runs (write output in space below).
- ☐ Compile-time error (explain source of error).
- ☐ Run-time error (explain source of error).

Compile-time error. Class Animal cannot be instantiated.

(e)
```
List<Animal> a5 = new ArrayList<Dog>();
a5.add(new Dog("Moby"));
a5.get(0).printName();
```

- ☐ Compiles and runs (write output in space below).
- ☐ Syntax error (explain source of error).
- ☐ Run-time error (explain source of error).

Compile-time error. Incompatible types in line 1.

(f)
```
Animal a6 = new Dog("Moby");
Dog d6 = (Dog) a6;
a6.printName();
d6.printName();
```

- ☐ Compiles and runs (write output in space below).
- ☐ Syntax error (explain source of error).
- ☐ Run-time error (explain source of error).

Compiles and runs. Output is:

```
Dog Moby
Dog Moby
```

Part 2: The following code segment correctly creates and populates a set of animals.

Segment 1

```
Set<Animal> zoo = new HashSet<Animal>();
Cat c1 = new Cat("Astor");
zoo.add(c1);
Cat c2 = new Cat("Nutmeg");
zoo.add(c2);
Dog d1 = new Dog("Moby");
zoo.add(d1);
Dog d2 = new Dog("Abby");
zoo.add(d2);
```

a) [6 marks] Assume that Segment 2 (shown below) immediately follows Segment 1. In the space below, indicate whether Segment 2 will compile correctly or not. If it compiles, then indicate what will happen at run-time. If it does not compile, explain why.

Segment 2

```
Iterator<Animal> it1 = zoo.iterator();
for (;it1.hasNext();)
{
  it1.next().printName();
}
```

Compiles ok (3 marks) and produces the output on the next page (3 marks).

b) [6 marks] Assume that Segment 2 has been removed and Segment 3 (shown below) immediately follows Segment 1. It compiles and runs with no errors.

Segment 3

```
Iterator<Animal> it2 = zoo.iterator();
for (;it2.hasNext();)
{
  Animal a = it2.next();
  if (a instanceof Cat)
  {
    Cat c = (Cat) a;
    c.printName();
  }
  if (a instanceof Dog)
  {
    Dog d = (Dog) a;
    d.printName();
  }
}
```

The output produced is:

```
Dog Abby
Dog Moby
Cat Astor
Cat Nutmeg
```

Assuming this is the correct output, provide a critique of Segment 3 (you may use the next page as well). How could it be improved? (Do not provide code, only a description of the changes you would make).

Does not use polymorphism (3 marks). Should really be replaced by Segment 2 (3 marks).

Q3. [30 marks] The UML diagrams of the following classes contain all their attributes. The default constructor for every class initializes all primitives to `0`, and all non-primitives to `null`. The functionality of methods `shallowCopy` and `deepCopy` is as their name suggests.

Ant
+ i : int + b : Bear
+ shallowCopy(): Ant + deepCopy(): Ant

Bear
+ d : double + c : Cat
+ shallowCopy(): Bear + deepCopy(): Bear

Cat
+ l : long + f : float
+ shallowCopy(): Cat + deepCopy(): Cat

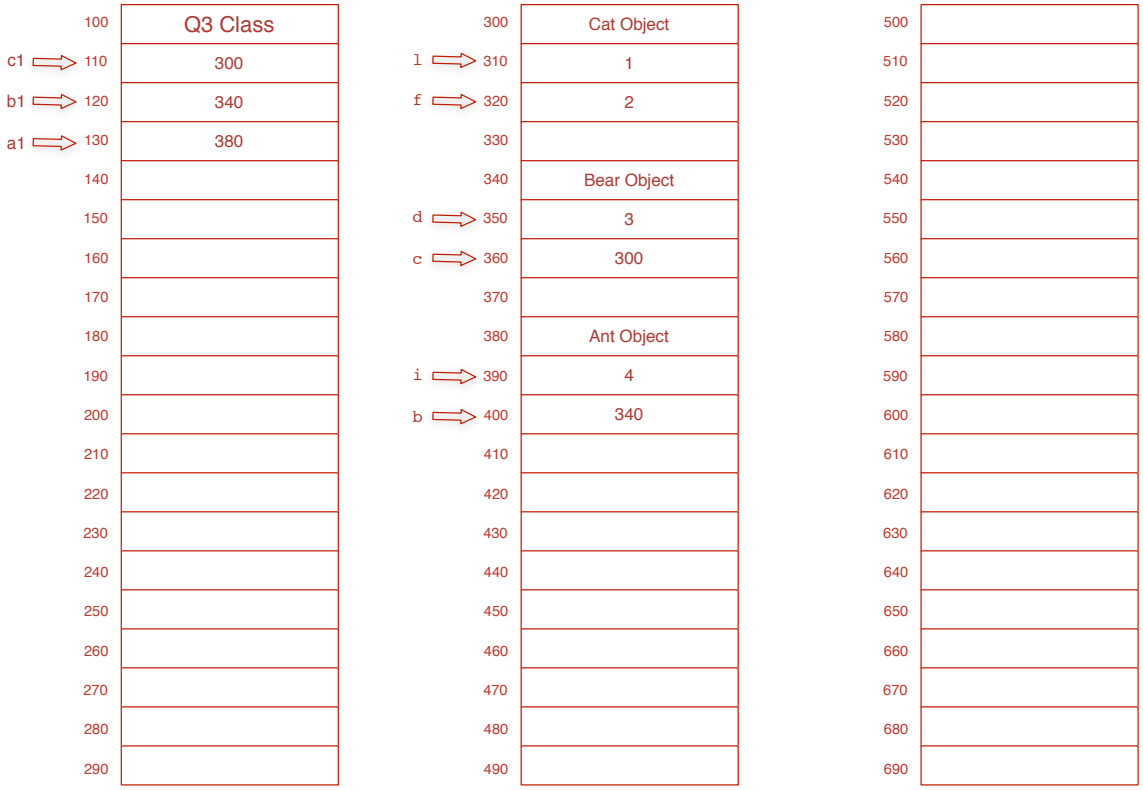
Consider the following app that uses these classes.

```
public class Q3 {
   public static void main (String[] args)  {
                Cat c1 = new Cat();
                c1.l = 1;
                c1.f = 2;
                Bear b1 = new Bear();
                b1.d = 3;
                b1.c = c1;
                Ant a1 = new Ant();
                a1.i = 4;
                a1.b = b1;
                // Draw diagram 1
                Ant a2 = new Ant();
                a2.i = 5;
                a2.b = b1;
                Cat c2 = new Cat();
                c2.l = 6;
                c2.f = 7;
                b1.c = c2;
                // Draw diagram 2
                Ant a3 = a2.shallowCopy();
                // Draw diagram 3
                Ant a4 = a3.deepCopy();
                // Draw diagram 4
                b1 = a4.b.shallowCopy();
                // Draw diagram 5
   }
}
```

In the following pages, draw memory diagrams to reflect the contents of memory at the five points during the execution of this app designated by the comments in the code. For diagrams 2-5, you don't need to redraw fully the parts that are unchanged from the previous diagram, but indicate these unchanged parts clearly. For each diagram, you can assume that the garbage collector has just run. You do not need to show parts of the memory where class definitions are loaded.

Tip: Reserve the first column below for the memory diagram of class Q3. Place all objects in the other two columns.

(a) [6 marks] Draw diagram 1.

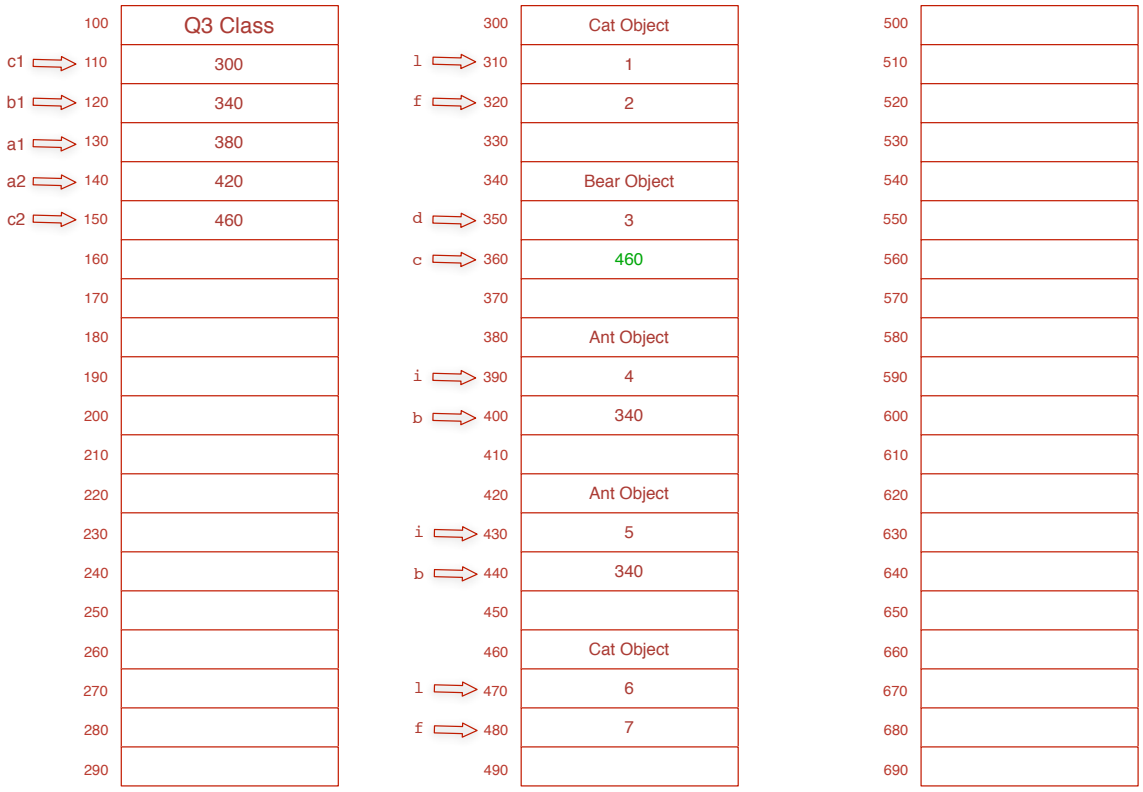


The address values do not have to be the same as above, but they have to point to the correct object. The primitive values must be exactly as shown.

-2 marks per missing or extra object

-1 mark per missing or incorrect value

(b) [6 marks] Draw diagram 2.

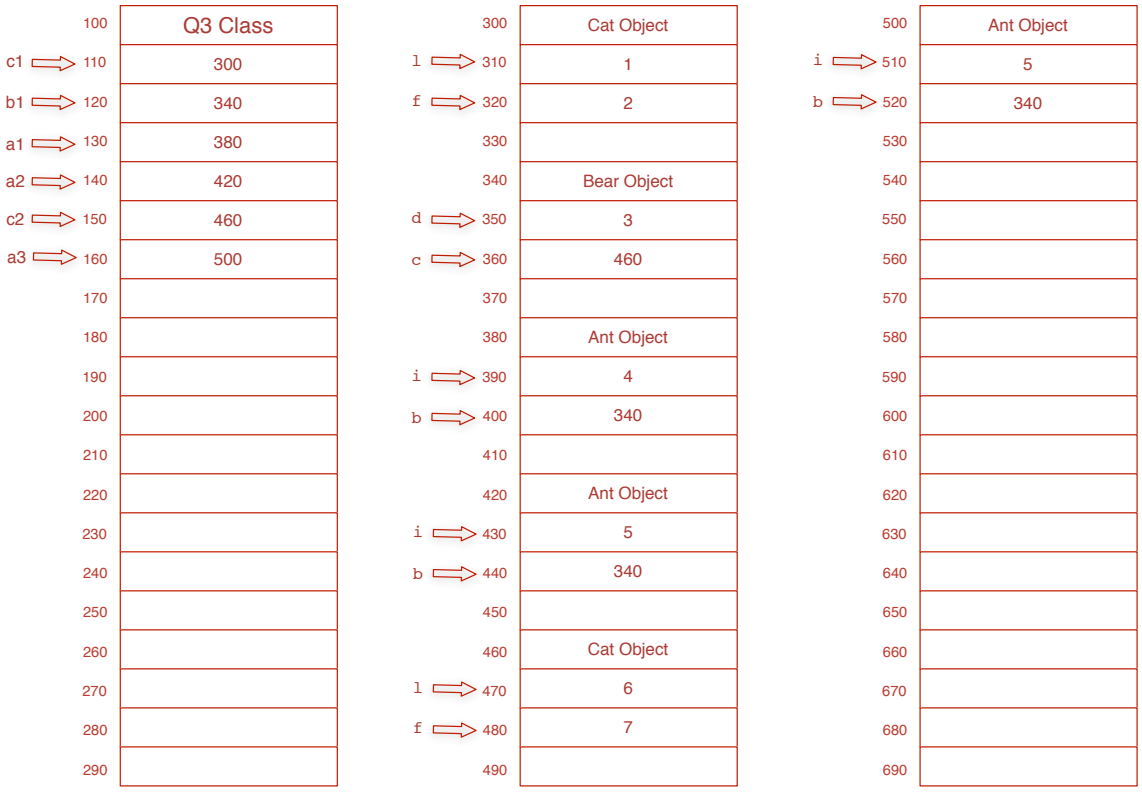


The address values do not have to be the same as above, but they have to point to the correct object. The primitive values must be exactly as shown. Changed values are shown in green. No objects are removed by the garbage collector.

-2 marks per missing or extra object

-1 mark per missing or incorrect value

(c) [6 marks] Draw diagram 3.

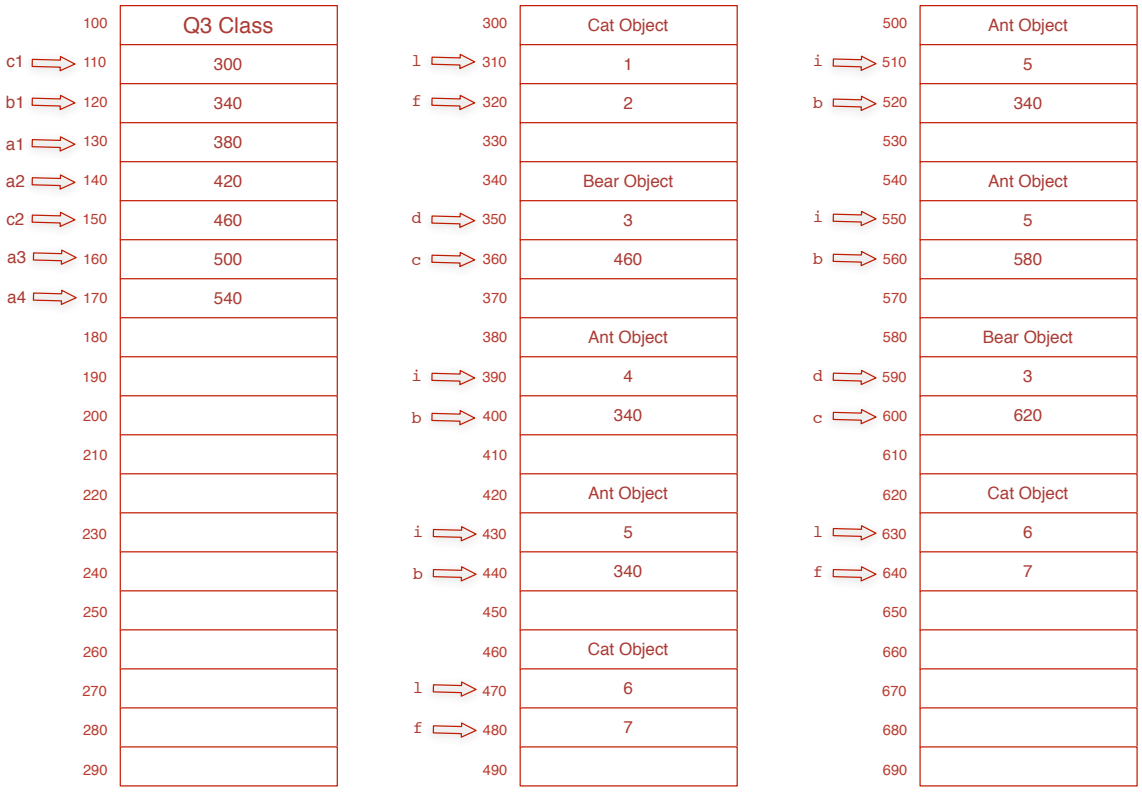


The address values do not have to be the same as above, but they have to point to the correct object. The primitive values must be exactly as shown. No objects are removed by the garbage collector.

-2 marks per missing or extra object

-1 mark per missing or incorrect value

(d) [6 marks] Draw diagram 4.

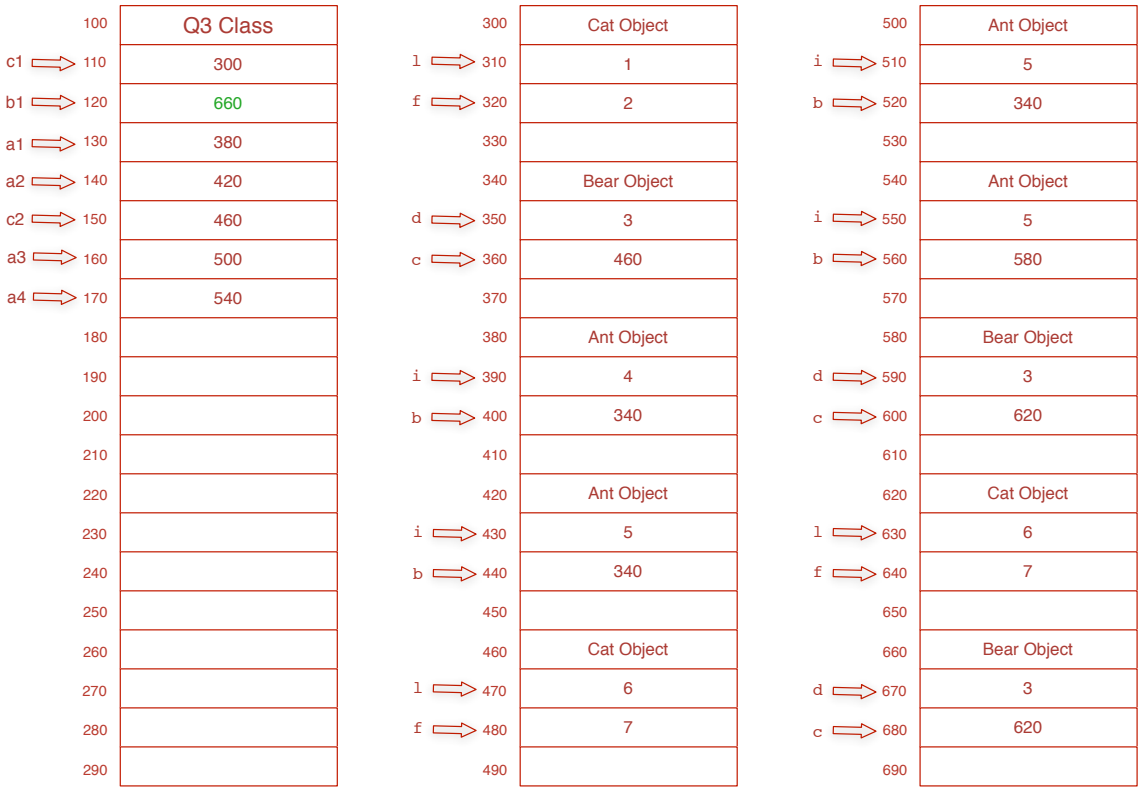


The address values do not have to be the same as above, but they have to point to the correct object. The primitive values must be exactly as shown. No objects are removed by the garbage collector.

-2 marks per missing or extra object

-1 mark per missing or incorrect value

(e) [6 marks] Draw diagram 5.



The address values do not have to be the same as above, but they have to point to the correct object. The primitive values must be exactly as shown. Changed values are shown in green. No objects are removed by the garbage collector.

-2 marks per missing or extra object

-1 mark per missing or incorrect value