Question - 1

Reversing Numbers

SCORE: **10 points**

Given an input integer, return a **reversed** value of the input.

Eg:
input -> 566788
output -> 887665

Question - 2

String

SCORE: **10 points**

Examine this code:

```
String myString = "";
```

What value is contained in `myString`?

- null
- String reference
- an empty reference
- a character reference

Question - 3

References

SCORE: **10 points**

When an object no longer has any reference variables referring to it, what happens to it?

- The garbage collector makes the memory it occupies available for new objects.
- It is swapped out to disk.
- It becomes spam.
- it sits around in main memory forever.

Question - 4

Reference Pointer

SCORE: **10 points**

if a method assigns a new *object* to an object reference parameter, will this have any effect on its caller?

- No, because this is not a legal operation.
- Yes, because now the caller can reference the new object.
- No, because this will not affect any object that the caller can reference.
- Yes, because the new object will replace one of the caller's objects.

Question - 5

SCORE: **10 points**

Alter String

Examine the following:

```
String mess = "Hello" ;
mess = mess + " World" ;
```

What does the second statement do?

- It creates a new String object based on the original object referenced by mess and another String object containing World.
- It is illegal because it attempts to alter an immutable object.
- It alters the immutable object referenced by mess.
- It adds the characters World to the String object referenced by mess.

Question - 6

SCORE: **10 points**

```
class Value
{
    public int i = 15;
}
public class Test
{
    public static void main(String argv[])
    {
        Test t = new Test();
        t.first();
    }
    public void first()
    {
        int i = 5;
        Value v = new Value();
        v.i = 25;
        second(v, i);
        System.out.print(" "+v.i+" ");
    }
    public void second(Value v, int i)
```

```
    {
        i = 0;
        v.i = 20;
        Value val = new Value();
        v =  val;
        System.out.print(v.i + " " + i);
    }
}
```

- (●) 15 0 20
- ( ) 15 0 25
- ( ) 20 0 25
- ( ) 20 0 20

Question - 7

SCORE: **10 points**

```
Given the code below, and making no other
changes, which access modifiers (public,
protected or private) can legally be placed
before myMethod() on line 3? If line 3 is left as
it is, which keywords can legally be placed
before myMethod on line 8?

1.  class HumptyDumpty
2.  {
3.      void myMethod() {}
4.  }
5.
6.  class HankyPanky extends HumptyDumpty
7.  {
8.      void myMethod() {}
9.  }
```

- ( ) public or protected on line 3. private or nothing (i.e. leaving it as it is) on line 8.
- (●) private or nothing (i.e. leaving it as it is) on line 3. Nothing (i.e. leaving it as it is) or protected or public on line 8.
- ( ) nothing (i.e. leaving it as it is) or protected or public on line 3. private or nothing (i.e. leaving it as it is) on
- ( ) None of the above

Question - 8

SCORE: **10 points**

Java

The package _____ is automatically imported in every java program.

- java.util
- java.lang.*
- java.util.*
- both B & C

Question - 9

SCORE: **20 points**

Insertion in Array

Given a sorted array and a target array, return an array of index for each target element in the final array if they were to be placed in the sorted array preserving the sorted order.

Eg:
sorted array -> [1, 7, 15, 79, 93]
target array -> [9, 28, 33, 92]
answer -> [2, 4, 5, 7] as the final array would look something like [1, 7, 9, 15, 28, 33, 79, 92, 93]

Assumptions :
1. there are no duplicates.
2. elements are always sorted in ascending order.