**CMSC 203 Lab 5**

**Memory Mapping**

1. Draw the memory map of the following one-dimensional array of type int.

int[] x = new int[8];

Stack: Heap:

0 0 0 0 0 0 0 0

**x[0] x[1] x[2] x[3] x[4] x[5] x[6] x[7]**

**x**

1. Draw the memory map of the following two-dimensional ragged array of ints.

int[][] y = {{4, 8, 15}, {16, 23, 42, 10}, {8, 30}};

Stack: Heap:

**15**

**8**

**4**

**16 23 42 10**

**8 30**

**y**

1. Draw the memory map of the following one-dimensional array of type String.

String[] z = new String[4];

for (int i = 0; i < z.length; i++) {

z[i] = "element " + i;

}

Stack: Heap:

**Z[0] z[1] z[2] z[3]**

**z**

1. a. Write a shallow copy of the following in code. (Assume the five animal objects are already instantiated.)

Animal[] zoo = {tiger1, elephant2, giraffe3, monkey4, ape5};

Animal[] copy;

Solution:

**Animal[] zoo = {tiger1, elephant2, giraffe3, monkey4, ape5};**

**Animal[] copy = new Animal[zoo.length];**

**for (int i = 0; i < zoo.length; i++) {**

**copy[i] = zoo[i];**

**}**

1. Draw the memory map.

Stack: Heap:

**zoo[0] zoo[1] zoo[2] zoo[3] zoo[4]**

**copy**

**copy[0] copy[1] copy[2] copy[3] copy[4]**

**zoo**

1. Write the deep copy of the following in code.

Animal[] zoo = {tiger1, elephant2, giraffe3, monkey4, ape5};

Animal[] copy;

Solution:  
  
**Animal[] zoo = {tiger1, elephant2, giraffe3, monkey4, ape5};**

**Animal[] copy = new Animal[zoo.length];**

**for (int i = 0; i < zoo.length; i++) {**

**copy[i] = new Animal(zoo[i]);**

**}**

1. Draw the memory map.

Stack: Heap:

**zoo[0] zoo[1] zoo[2] zoo[3] zoo[4]**

**copy**

**copy[0] copy[1] copy[2] copy[3] copy[4]**

**zoo**

1. What is garbage collection? Where does it happen?

Garbage collection is an automatic process in programming languages that identifies and removes unused or unreferenced objects from memory to free up space. It occurs within the memory management system of a programming language, cleaning up allocated memory that's no longer in use.

1. What is the difference between the two operators, equals() and ==?

The **equals()** method is used to compare the content or values of objects, while the **==** operator checks if two objects share the same memory location (for objects) or compares actual values (for primitive types).