*CSE 102*

**ArrayList and Vector**

A Quick Recap:

* Vector and ArrayList classes are almost equivalent, but ArrayList is usually preferred.
* An array is simply a contiguous memory to store data plus its length information stored elsewhere, while ArrayList and Vector are fully-fledged classes which was implemented using regular arrays. They include methods to access, add, and delete elements as well as private methods (that cannot be directly called from outside) to resize the underlying array whenever needed.
* It is possible to store a primitive value directly in an array while ArrayList can only hold references. Therefore you need to use wrapper classes for primitive types (Integer for int, Character for char etc.)
* We prefer ArrayList over an ordinary array when we don’t know the size information beforehand or when the size needs to be changed dynamically during the execution of the program.
* There is no special syntax for a multi-dimensional ArrayList, you can create it via nesting (e.g. ArrayList<ArrayList<Integer>>).

**Exercises**:

1. The following code is supposed to store numbers entered by user in memory until the user enters 0. Which statement should be put in place of the comment?

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

Scanner sc = **new** Scanner(System.***in***);

ArrayList<Integer> nums = **new** ArrayList<>();

**int** cur;

**do** {

cur = sc.nextInt();

// Here

} **while**(cur != 0);

}

1. nums[i] = cur;
2. nums.add(cur);
3. nums = nums.add(cur);
4. nums.append(cur)
5. What will be the output of the following program?

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

ArrayList<Integer> nums = {2,3,5};

System.***out***.println(nums.get(0));

}

1. 0
2. 2
3. Runtime Error
4. Compiler Error
5. What will be the output of the following program?

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

ArrayList<Integer> nums = **new** ArrayList<>();

nums.add(3);

nums.add(5);

nums.add(nums.get(0));

System.***out***.println(nums);

}

1. [3, 5, 3]
2. [3, 3, 5]
3. [3, 5]
4. Something like [I@1c4af82c
5. In which of the following scenarios we need a dynamically resizable storage?
6. Storing integer values received from user where the first value entered is assumed to be the number of values following.
7. Storing integer values provided in the source code.
8. Storing integer values which were accessed through an array parameter in a function.
9. Storing integer values received from user until user enters a specific value.
10. What will be the output of the following program?

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

Vector<Integer> v = **new** Vector<>(0,10);

v.add(4);

System.***out***.println(v.capacity() - v.size());

}

1. 1
2. 0
3. 9
4. 4
5. What will be the output of the following program?

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

ArrayList<Integer> arr = **new** ArrayList<>();

arr.add(4);

System.***out***.println(arr.capacity() - arr.size());

}

1. 1
2. Compiler Error
3. 4
4. 9
5. Write a method which takes two ArrayList of integers and returns their intersection (the elements that appear in both lists) in the order they appear in the first list.
6. Write a method which takes an ArrayList of integers and removes any duplicate elements in it.

**ANSWERS:**

1. B
2. D
3. A
4. D
5. C
6. B
7. **static** ArrayList<Integer> intersection(

ArrayList<Integer> a,

ArrayList<Integer> b)

{

ArrayList<Integer> res = **new** ArrayList<>();

**for**(**int** i=0; i<a.size(); i++)

**if**(b.contains(a.get(i)))

res.add(a.get(i));

**return** res;

}

1. **static** **void** uniqify(ArrayList<Integer> a) {

**for**(**int** i=0; i<a.size(); i++)

**if**(a.lastIndexOf(a.get(i)) != i)

a.set(i, **null**);

**while**(a.remove(**null**));

}

Alternatively:

**static** **void** uniqify(ArrayList<Integer> a) {

**for**(**int** i=0; i<a.size(); i++)

**if**(a.lastIndexOf(a.get(i)) != i)

a.remove(i--);

}

Consider why we decrement i when we remove an element