Name: _____

USC Student ID Number: _____

USC NetID (e.g., `ttrojan`): _____

CS 455 Midterm Exam 1
Fall 2022 [Bono]

Tuesday, Sept. 27, 2022

There are 5 problems on the exam, with 65 points total available. There are 10 pages to the exam (5 pages **double-sided**), including this one; make sure you have all of them. If you need additional space to write any answers or scratch work, pages 9 and 10 are left mostly blank for that purpose. If you use those pages for answers you just need to direct us to look there. ***Do not detach any pages from this exam.***

Any remote students printing the exam should scan, and submit all pages of the exam, even if you didn't write an answer on a particular page. If you print it single-sided, do not write on the backs of pages and do not scan blank backs of pages.

Note: if you give multiple answers for a problem, we will only grade the first one. Avoid this issue by labeling and circling your final answers and crossing out any other answers you changed your mind about (though it's fine if you show your work).

Put your name, USC Student ID, and USC username (a.k.a., NetID) at the top of the exam. Also put your NetID at the top right of the front side of each page of the exam (including the last page). Please read over the whole test before beginning. Good luck!

`Arrays.toString` method documentation (adapted from Oracle documentation):

`public static String toString(int[] a)`

Returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets (`"[]"`). Adjacent elements are separated by the characters `", "` (a comma followed by a space). Returns `"null"` if a is `null`.

Problem 1 [5 points]

Consider the following incomplete program to finds the first vowel in a lower-case word. **Complete the loop condition for the program (fill in the box). Do not change anything else in the code.**

Note: The `charAt(index)` method used below allows you to access individual chars in a `String`. The indices are zero-based, as they are in arrays and ArrayLists.

```
public class FirstVowel {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      String word = in.next();
      boolean found = false;
      int position = 0;

      while (                                                    ) {
         char ch = word.charAt(position);
         if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch =='u') {
            found = true;
         }
         else {
            position++;
         }
      }
      if (found) {
         System.out.println("First vowel: " + word.charAt(position));
         System.out.println("Position: " + position);
      }
      else {
         System.out.println("No vowels");
      }
   }
}
```

Problem 2 [7 pts. total]

Consider the following program:

```
public class Prob2 {
   public static void main(String[] args) {
      int[] myNums = new int[3];
      initArray(myNums, 6);
      System.out.println(Arrays.toString(myNums));
   }

   public static void initArray(int[] nums, int size) {
      nums = new int[size];
      for (int i = 0; i < nums.length; i++) {
         nums[i] = i;
      }
   }
}
```

Part A [5]. In the space below, draw a box-and-pointer diagram (a.k.a., memory diagram) showing all object variables, objects, and their state as they have changed during the code sequence. This includes showing `initArray`'s parameters.

Part B [2]. What is printed by the code? (Documentation for `Arrays.toString` appears on the exam cover page.)

Problem 3 [29 pts. total]

Part A [13]. Complete the implementation of the class `VendingMachine`, described further below, and defined on this and the next page.

A `VendingMachine` models the inventory inside a vending machine that can hold three types of candy. A client can buy candy, find out how much candy there is, and supply the machine with more candy. (Note: The money part of the transactions are not part of this class.) Here is an example of the use of this class:

```
public static void main(String[] args) {
   VendingMachine vend = new VendingMachine();  // machine is empty
   System.out.println(vend.buy(VendingMachine.HERSHEYS, 3));
                                // no product in the machine, machine gives 0
   vend.supply(VendingMachine.HERSHEYS, 20);    // stock the machine
   vend.supply(VendingMachine.M_AND_MS, 20);
   vend.supply(VendingMachine.STARBURST, 20);
   System.out.println(vend.buy(VendingMachine.HERSHEYS, 3));
                                  // customer buys 3 Hersheys bars (prints 3)
   System.out.println(vend.buy(VendingMachine.HERSHEYS, 18));
                  // customer attempts to buy 18 Hersheys, only receives 17
   System.out.println(vend.buy(VendingMachine.HERSHEYS, 3));     // prints 0
   System.out.println(vend.get(VendingMachine.HERSHEYS));// How many left? 0
   vend.supply(VendingMachine.HERSHEYS, 5); // ran out of HERSHEYS, add more
   System.out.println(vend.buy(VendingMachine.STARBURST, 1));    // prints 1
   System.out.println(vend.buy(VendingMachine.HERSHEYS, 10));    // prints 5
   . . .
}
```

The class interface and method comments appear below and continue onto the next page. We left space so you can add in your code to complete the class:

```
//   VendingMachine models the inventory inside a candy machine.
//   interface invariant: the amount of each kind of candy in the machine
//                        is always non-negative
public class VendingMachine {
   public static final int HERSHEYS = 0;
   public static final int M_AND_MS = 1;
   public static final int STARBURST = 2;
   public static final int NUM_KINDS = 3;









   //   Creates an empty vending machine object
   public VendingMachine () {









   }
```

Problem 3 (cont.)

```
   // PRECONDITION for parameters to get, buy, and supply methods:
   // -- kind is one of HERSHEYS, M_AND_MS, and STARBURST
   // -- quantity > 0   [applies to buy and supply only]

   // Returns the amount in this machine of the kind of candy specified
   // PRECONDITION: see above
   public int get(int kind) {









   }

   //  Adds to this vending machine the given quantity of the kind of candy
   //     specified.
   // PRECONDITION: see above
   public void supply(int kind, int quantity) {









   }

   // Attempts to buy the given quantity of the kind of candy specified
   // from this vending machine.  Returns the actual number of pieces
   // given out, based on the current inventory (i.e., it can't give out
   // more than is currently in the machine); updates inventory accordingly.
   // PRECONDITION: see above
   public int buy(int kind, int quantity) {









   }
}
```

Problem 3 (cont.)

Part B [3]. Suppose we now want to add a fourth type of candy in our `VendingMachines`. What parts of your `VendingMachine` class code would have to change to handle the fourth candy type? Circle all that apply below (we started it for you):

(a.) comments

b. constant definitions

c. instance variable definitions

d. constructor

e. `get` method

f. `buy` method

g. `supply` method

Part C [6]. Write a *representation invariant* for your `VendingMachine` class below:

Note: Parts D and E are concerned with using the `VendingMachine` class.

Part D [2]. Show the variable definition(s) for a data structure that would allow us to store information about *all* the `VendingMachines` in a building:

Part E [5]. Give the code to initialize your data structure from part D so that there are 10 empty vending machines in the building, or say "already done", if your data structure was already implicitly initialized this way by the code you gave for part D.

USC NetID: _____

Reminder of some Linux shell commands and other shell syntax (for problem 4 below):
`ls  cd  cp  mv  rm  mkdir  rmdir  more  diff  ~  *  ..  <  >  .`

Problem 4 [4 points]

The directory in Vocareum that has all the lecture code examples is `$ASNLIB/public` in the Vocareum *Lecture code* item. Suppose you are in your workspace for that item. Write a sequence of one or more Linux commands to copy *all* the files inside the `08-28` subdirectory of the lecture code examples directory to your home directory.

Relevant information:

- You may assume there are no subdirectories in the Lecture Code `08-28` directory, just regular files.
- `$ASNLIB` is a shell variable defined on Vocareum that expands to the correct *absolute* path to get to that specific directory.
- A reminder of the names of some common Linux commands and related syntax appears above this problem.
- The commands below (using $ as the Linux prompt) show the starting conditions in your Lecture code workspace, including the current contents of that workspace.

```
$ cd
$ ls
$
```

Problem 5 [20 points]

Implement the static `zeroesRemoved` method, which takes an array of ints, `nums`, and returns an array with the same values in the same order as `nums`, but with all the zeroes removed. The method does not change the `nums` array. For full credit you may not use any extra arrays or other Java objects: only the two that are the parameter and return value of the function.

Here are some examples:

`nums`	**`zeroesRemoved(nums)`**
`[-1, -1, -4]`	`[-1, -1, -4]`
`[3, 14, 7, 0, 0]`	`[3, 14, 7]`
`[2, 0, 4, 0, 0, 0, 4, 5, 0, 3, 9]`	`[2, 4, 4, 5, 3, 9]`
`[0, 0, 0]`	`[]`

```
public static int[] zeroesRemoved(int[] nums) {
```

Extra space for answers or scratch work.

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.

Extra space for answers or scratch work (cont.)

If you put any of your answers here, please write a note on the question page directing us to look here. Also label any such answers here with the question number and part, and circle the answer.