STOP: Before you go any further, write your name and student ID on the front of each page of this exam, three times in total! You MUST DO THIS, as we may not be able to give you credit on unlabeled pages!

The exam period is two hours, though it is expected to take most students about one hour to complete. This exam has a total of 100 points. Note how many points each question is worth, and plan your time accordingly.

Clearly indicate your final answer to each question. If you do not, we will consider the first possible value in your answer as your final answer. Full credit will be awarded for correct answers with no additional incorrect information. Partial credit might be awarded for incomplete or partially correct answers.

For all coding questions, your code must not contain unnecessary method calls or loops, even if they do not otherwise impact correctness, and it may not import or use classes or methods not mentioned in the question. If you are asked to write or use methods that include genericity (that is, type parameters), your code must fully support generics.

This exam is closed book and closed notes: no supplementary material is allowed. You may not use an electronic device, such as a laptop or cellphone. No collaboration between students is permitted. Violating these rules or any portion of the University's Academic Honesty Policy is academic dishonesty.

Name and Student ID: _____

1. (5 points) Read the following Java method, and write a concise, high-level, English description of its result — the what, not the how. If you mechanically describe its action, you will lose significant credit.

```
int[] mystery(int[] a) {
    int[] b = new int[a.length];
    int j = a.length - 1;
    for (int i: a) {
        b[j] = i;
        j--;
    }
    return b;
}
```

2. (5 points) Identify the logical error in the following code. The error is conceptual (that is, the code does not work as described), and not a syntactic error (that is, an error the compiler would identify).

```
/**
 * Swaps the `i`th and `j`th elements of the array `a`.
 * `a` is guaranteed to have length >= max(i, j) - 1
 */
void swap(int[] a, int i, int j) {
    a[i] = a[j];
    a[j] = a[i];
}
```

3. (10 points) A palindrome is a string that reads the same backward and forward, for example, "abba" or "amanaplanacanalpanama". Write a method `static boolean isPalindrome(String str)` that returns `true` if and only if the value of the parameter is a palindrome. The `charAt` method of `String` may be helpful. Assume the value of the parameter `str` is a lowercase string containing only alphabetic characters.

   ```
   static boolean isPalindrome(String str) {
   ```

4. (10 points) Write a method `static <E> List<E> flatten(List<List<E>> lists)`. Given an input representing a list of lists, `flatten` should return a single list containing the elements from each list in the input, in order. For example, on input [[1, 2, 3,], [5, 6], [4]], `flatten` should return [1, 2, 3, 5, 6, 4]. `flatten` must not modify the input list.

   You may assume `List` and `ArrayList` are correctly imported from `java.util`.

   ```
   static <E> List<E> flatten(List<List<E>> lists) {
   ```

Name and Student ID: _____

5. (10 points) Write a generic class `MySet<E>` that extends an existing implementation of `Set`. `MySet` should include a public instance method called `difference`, which takes a single `Set<E> other` parameter. `difference` returns a new set consisting of the set theoretic difference between the other set's contents and the current set's contents, in that order. The set theoretic difference between two sets A and B, written as A - B, consists of all the elements of A that are not in B.

   `difference` must not modify either set. You may use `Set` and `HashSet` but no other classes from the `java.util` namespace.

6. State the approximate worst-case running time for each of the following.

   Appropriate answers are of the form "constant time" or "X in Y", where X might be "linear," "quadratic," or the like, and Y is the name of the variable or quantity in question. For example, a traversal of an array is linear in the length of the array; if the array is named `a`, it is linear in `a.length`.

   (a) (3 points) `HashMap.size`

   (b) (3 points) `ArrayList.indexOf`

   (c) (4 points)

```
int crossProductIsh(int[] a) {
  int sum = 0;
  for (int i = 0; i < a.length; i++) {
    for (int j = 0; j < a.length; j++) {
      sum += a[i] + a[j];
    }
  }
  return sum;
}
```

7. (10 points) Write a method `static <K, V> void removeFromMultiMap(Map<K, List<V>> map, K key, V value)`. `removeFromMultiMap` must remove the `value`, if present, from the list, if present, associated with the given `key`. If either is not present it should return without raising an exception.

   You may assume `Map`, `List`, `HashMap`, and `ArrayList` are correctly imported from `java.util`.

   ```
   static <K, V> void removeFromMultiMap(Map<K, List<V>> map, K key, V value) {
   ```

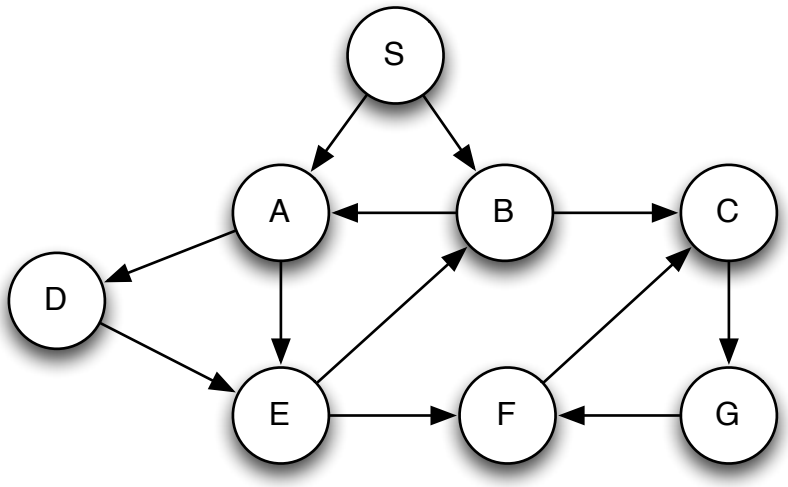
8. (4 points) Suppose you have an empty queue of integers, and on that queue you execute the following operations:

   - add 1
   - add 2
   - remove
   - add 2
   - add 3
   - add 1
   - remove
   - add 2
   - remove

   Show the final contents of the queue. Clearly indicate the front and back of the queue.

Name and Student ID: _____

9. (10 points) Suppose you have the following directed graph.



If you were to perform a breadth-first search to completely explore this graph, starting from S, in what order would the vertices be added to the frontier? Assume that the neighbors of each vertex are returned in alphabetical order. Remember to include S in your answer, and do not stop the search if it finds node G; completely explore the graph.

10. (10 points) Suppose you have an array-based implementation of a `Stack<E>`, which stores values in an instance variable `E[] array` and stores the index of the element at the top of the stack in an instance variable `int top`. Implement the peek method. You may assume other stack methods (`size`, `isEmpty`, `isFull`) are available if you need them. You may assume `StackUnderflowException` is correctly imported.

```
public E peek() throws StackUnderflowException {
```

11. (10 points) The `String` class has a `public String toLowerCase()` method, which returns a new string with all of the characters in the original instance of `String` converted to lower case. Write a unit test that verifies this method works correctly on a `String` containing at least one upper and one lower case letter.

    You may assume the `String` and `Test` classes, and the `assertEquals` static method are all correctly imported.

12. (6 points) Suppose you wish to compute the product of the integers $1 \ldots n = 1 \times 2 \times 3 \times \ldots \times n$. Write a recursive method to do so. Assume $n > 0$.

```
static int product(int n) {
```