HW2 Grading Rubric

Written 1 (10 Points)

- (+5) Test with L = {1, 11, 15} and P = {0,6}
  - -1 For each wrong element
  - -5 if doesn't work at all, printing out wrong list
- (+1) Code is syntactically correct Java
- (+3) If solution uses Collection API correctly
  - Using methods of collection API correctly
- (+1) Checks for inequality in length of Lists
- (-3) if solution uses any List specific methods
  - Using array indices, using methods in Arrays class

Written 2 - 3.4 (10 Points)

- (+5) Test with L1 = {-100, 0, 1} and L2 = {-100, -1, 0, 100}
  - (-1) per extra element or missing element from {-100, 0}
- (+3) Code:

  (+1) terminates appropriately,

  (+2) either prints out the intersection or returns it as a collection/list
- (+2) If code is Java-like ("Perform binary search" is not java like, "Collections.binarySearch" is ok)
- (-4) Code does not use iterators and instead repeatedly uses get(x), or some other strategy that is comparatively much weaker in terms of runtime complexity, e.g O(n^2)
  - (-2) for code that uses iterators but still O(n^2), i.e no better than a naive implementation
  - (-2) for code that doesn't use iterators, but better than O(n^2), e.g O(nlogn) via binary search
- (-1) uses toArray
- (-3) if solution uses any operation other than basic list operations - e.g using linkedlist next instead of iterator next
- (Not marked) Handles duplicates - although strictly speaking, sets should not contain repetitions

Written 3 - 3.24 (10 points)

- (-5) total for missing methods:
  - -1 if TwoStack class/constructor method is missing
  - -1 if push1 or push2 is missing
  - -1 if pop1 or pop2 is missing
  - (-1 if the above is not explicitly written in java like pseudocode)
- (-4) total for correctness of methods. -1 for each of the following parts that is not working (e.g. does not check for stack overflow, stack pointer is not moved, etc.).
  - Push1
  - Push2

    - Pop1
    - Pop2
- (-1) Inefficient or incorrect algorithm (eg. moving the stacks within the array when pushing or popping, copying from array to actual stacks)
- (-1) Does not support generics
- (-1) Uses data structure other than array
- (-1) Has single push/pop method with no parameter for which stack

Written 4 (10 points)

a) (7)

- (-4) if solution utilizes more than the 3 holding tracks
- (-2) if proposed algorithm does not have steps listed in correct order or is missing a small intermediary step; also -2 if only one step is incorrect
- If solution, though incorrect, is given with explanation is rational, then give 3 points for partial credit (or if solution is incomplete: too vague or missing a lot of steps)

b) (3)

- If incorrect sequence provided, but explanation or rationale given, then give 1 point
- If incorrect sequence provided and no work, then no credit given

All Programming Problems

Compiling Issues:

-2 Simple Fix to Compile

-4 Slightly more complicated issue

Major Issue: Max ⅔ of total grade based on attempt

Programming 1 (30 Points)

- (-2) If Test.java doesn’t cover an error condition
- (-2) If not following instructions

MyStack (7 points)

- (+2) O(1) pop
- (+2) O(1) push
- (+3) Correct implementation that doesn’t use push, pop of linked list

Test (16 points)

- (+2) For each test case that passes

SymbolBalance.java (7 points)

- (+3) Uses the Stack properly
- (+2) Uses command line arguments
- (+2) Correct File I/O

Programming 2 (30 points)

TwoStackQueue (19)

- (+6) correct implementation of enqueue()
- (+6) correct implementation of dequeue()
    - (-3) for inefficiency of moving back/forth
    - (-1) Runtime exception when dequeuing empty stack that is not NoSuchElement
- (+2) correct usage of two stacks (using MyStack class from PP1)
- (+2) correct usage of generics
- (+1) file is correctly called TwoStackQueue
- (+2) Implements the given interface(MyQueue.java)

Program2.java (7)

- (+2) correct usage of enqueue()
- (+2) correct usage of dequeue()
- (+2) output is printed correctly to screen
- (+1) file is correctly called Program2

Program2.txt (4)

- (+2) adequate discussion of Big-O of enqueue()
    - (-1) if wrong but has reasonable explanation
- (+2) adequate discussion of Big-O of dequeue()
    - (-2) if wrong/both enqueue and dequeue are O(1)
    - (-1) if O(N) and no explanation of O(1) when outbox has elements
    - Full points only if the answer is O(N) when outbox is empty, O(1) when outbox has elements
- (-1) If file doesn’t exist, but explanation is somewhere