CISC 3130 Ari Mermelstein

Midterm exam #2 Fall 2022

Question 1: This question is about sorting. (20 points)

A. Given the array:

```
int numbers[] = {30, 29, 7, 45, 40, 36, 49, 30};
```

Describe how to sort the array using insertion sort. Show what the array looks like after each pass.

B. Run the partition() method described in class to partition the array using 21 as the pivot.

```
21 4 33 48 12 4 16 40
```

C. Run the merge() method described in class to sort the two sorted subarrays. Please show all comparisons and what elements get copied.

6 9 23 50 1 6 9 32

D. Here are 2 definitions:

1. A sorting algorithm is said to be **inplace** if it is able to sort using only O(1) extra space.

2. A sorting algorithm is said to be **stable** if when 2 elements are equal, the one that started to the left in the array stays to the left and the one to the right stays to the right.

For bubblesort, insertion sort, merge sort and quicksort, explain if they are inplace or not, and explain if they are stable or not.

Question 2: This question is about priority queues (12 points)

A. What is printed by this program?

```
PriorityQueue < Integer >   pQueue;
pQueue = new PriorityQueue<>();
Integer[] numbers = {39, 7, 41, 30, 17, 29, 17, 20};
for(int x : numbers)
        pQueue.add(x);
for(int i=0; i<4; i++)
        System.out.println(Integer.toString(pQueue.poll()));
```

B. Design a class that implements the Comparator<Integer> interface as follows: The method that your class contains should return a negative number if the first number is larger than the second, a positive number if the first number is smaller than the second, and 0 if the numbers are the same.

C. Rerun the code from part A, but instead of declaring the priority queue using the default constructor, declare it using the comparator from part B instead.

Question 3: Our implementation of Priority Queues (20 points)

A. What are the running times of the following priority queue operations if we implement a priority queue using a binary min heap?

1. add()

2. poll()

3. peek()

4. isEmpty()

5. size()

B. Is the following array a legal binary min heap, yes or no? You must explain why it is a binary min heap, or if it isn't, you must explain why not.

```
int[] arr = {4, 5, 34, 14, 37, 28, 36, 14, 47};
```

C. Show what the array representation and tree representation of the heap looks like after you insert a 1 into the min heap.

```
int[] arr = {4, 7, 4, 15, 37, 5, 9, 42, 50, 50};
```

D. Show what the array representation and tree representation of the heap looks like after your remove the min,

```
int[] arr = {4, 7, 4, 15, 37, 5, 9, 42, 50, 50};
```

Question 4: Stacks and Queues (20 points)

A. Show what the data structures look like after each commented line. Clearly show what the top of the stack is and what the front and end of the queue are. Assume the stacks and queues we wrote in class. (8 points)

```
MyStack <Character > stack = new ArrayStack<>();

for(char ch = 'a'; ch<='e'; ch++)

             stack.push(ch);

//Comment 1

while(!stack.isEmpty())

          stack.pop();

//Comment 2

MyQueue <Integer > queue = new LinkedQueue <>();

for(int i=10; i<=15; i++)
            queue.add(i);

//Comment 3

for(int i=0; i<3; i++)

            queue.poll();

//Comment 4
```

B. Show how we would declare a Stack and a Queue of integers in standard Java. Show how to add the numbers 1-5 to the stack and queue. (4 points)

C. What are the running times for the following Stack operations? Explain why these are the running times by appealing to how we implemented these ourselves in class. (8 points)

1. push()

2. pop()

3. isEmpty()

4. peek()

Question 5: Problem Solving (20 points)

Solve 2 out of the following 3 problems. You must solve these problems as efficiently as possible. Each requires you to use a stat structure we described in class (e.g. Set, stack PriorityQueue).

1. Given a Stack and a target number inside the stack, decide whether or not the number is contained inside the stack. You must make sure that your algorithm doesn't modify the original stack permanently such that the stack looks untouched at the end.

```
public static boolean contains(ArrayStack<Integer> stack, int targer){

}
```

2. Given an array of integers, determine whether or not the array contains any duplicate elements. (Hint: this is similar to an interview problem we did)

```
public static boolean containsDuplicates(int[ ] arr) {



}
```

3. Given an array of integers, find the median (middle number) in the array. That is, find the number in the array that would be at index floor(n/2) in the sorted order. Do not sort!

```
public int findMedian(int [] arr) {




}
```

Question 6: Sets and Maps (8 points)

A. Briefly explain the difference between a HashSet and a TreeSet in terms of how the data is returned by the iterator. Describe one advantage to using a HashSet and one advantage of using a TreeSet.

B. Suppose that I wanted to implement a Set using a sorted array. Remember that a set doesn't allow duplicates. Briefly explain how you would implement add(x), remove(x), and contains(x) and analyze the worst case time for all of them.

E.C.1. Prove that the recurrence T(n) = 2 T(n/2) + n has solution O(n log n)

2. Prove that the height of a binary heap is O(log n).

3. Explain how to turn the array {9, 6, 10, 4, 3, 11} into a binary min heap by using the algorithm we described in class.

4. A binary max heap is analogous to a binary min heap, except that instead of the minimum always being at the root, the maximum is instead. Define the binary max heap property.