**COMP 2150 – Fall 2019**

**Homework** **8: BigO, Stack,Queue, Sort**

**( 92 points)**

Number of People: Individual. Feel free to ask me for help, or visit the Computer Science Learning Center (<http://www.memphis.edu/cs/current_students/cslc.php>).

Due: Nov 24 2019 by 11:59 pm

Submission: Zip all your Java source files (you can zip the entire project folder if using an IDE) into a single file and upload it to the proper folder in the eLearn dropbox at <https://elearn.memphis.edu>. **Do not include Password protection on you zipped file.**

Coding Style: Use consistent indentation. Use standard Java naming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT\_NAMES**. Include a reasonable amount of comments.

1. ( 10 points/2 points each) Use the Big O notation to estimate the time complexity of the following methods.

**public** **static** **void** mA(**int** n) {

**for** (**int** i = 0; i < n; i++) {

System.out.print(Math.random());

}

}

Ans: O(n)



**public** **static** **void** mB(**int** n) {

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < i; j++)

System.out.print(Math.random());

}

}

Ans: O(n2)

**public** **static** **void** mC(**int**[] m) {

**for** (**int** i = 0; i < m.length; i++) {

**for** (**int** j = 0; j < i; j++)

System.out.print(m[i] \* m[j]);

}

}

Ans: O(n2)



**public** **static** **void** mD(**int**[] m) {

**for** (**int** i = 0; i < m.length; i++) {

System.out.print(m[i]);

}

**for** (**int** i = m.length - 1; i >= 0; )

{

System.out.print(m[i]);

i--;

}

}

Answer: O(n)

**E)**

Int Sum = 0;

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 1; j <= i; j += 2)

sum += 4;

}

**for** (**int** k = - 100 ; k <= -1; k++)

sum--;

Ans: O(n log n)

1. **(10 points**) Design an O(n) time algorithm for computing the sum of numbers from n1 to n2 for (n1 < n2). Can you design an O(1) for performing the same task?

Ans:

int sum = 0;

for(int i = n1; I <= n2; i++){

sum++;

}

**Yes:**

int sum = (n2 \* (n2 + 1) / 2 – n1 \* (n1 – 1) / 2);

1. (8 points) Write the state of the elements of the following array after each pass of the outmost loop of the **selection sort** algorithm has occurred (after each element is selected and moved into place)

Int[] n1 = {63, 9, 45, 72, 27, 18, 54, 36}

{63, 9, 45, 27, 18, 54, 36, 72}

{9, 45, 27, 18, 54, 36, 63, 72}

{9, 45, 27, 18, 36, 54, 63, 72}

{9, 27, 18, 36, 45, 54, 63, 72}

{9, 27, 18, 36, 45, 54, 63, 72}

{9, 18, 27, 36, 45, 54, 63, 72}

{9, 18, 27, 36, 45, 54, 63, 72}

1. (8 points) Using the same array from Q3 above, trace the complete execution of the **merge sort** algorithm. Show the subarrays that are created by the algorithm and show the merging of subarrays in the larger sorted array.

Int[] n1 = {63, 9, 45, 72, 27, 18, 54, 36}

{63, 9, 45, 72, 27, 18, 54, 36}

{63, 9, 45, 72} {27, 18, 54, 36}

{63, 9} {45, 72} {27, 18} {54, 36}

{63} {9} {45} {72} {27} {18} {54} {36}

{9, 63} {45, 72} {18, 27} {36, 54}

{9, 45, 63, 72} {18, 27, 36, 54}

{9, 18, 27, 36, 45, 54, 63, 72}

**Queue and Stack questions**

1. (8 points / 4 points each) Show the output when the following integer stacks pass-through this static method:

You will need to write a java class with the main(String[] args) {….}, build a stack array, and run this doThis(….) method to show the output

1. [2, 6, 1]
2. [42. -3, 4, 15, 9]

Answers:

1. [1, 1, 6, 6, 2, 2]
2. [9, 9, 15, 15, 4, 4, -3, -3, 42, 42]

**public** **static** **void** doThis**(**Stack **<**Integer**>** x**)** **{**

Queue**<**Integer**>** q **=** **new** LinkedList**<>();**

**while(!**x**.**isEmpty**())** **{**

**int** n **=** x**.**pop**();**

q**.**add**(**n**);**

q**.**add**(**n**);**

**}**

**while** **(!**q**.**isEmpty**())** **{**

x**.**push**(**q**.**remove**());**

**}**

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

**}**

1. (8 points / 4 points each ) Show the output when the following queues integers pass through this static method:

You will need to write a class with the main(String[] args) {….} method, build a queue array and run this doThis(….) method to show the results

1. [1, 2, 3, 4, 5, 6]
2. [42. -3, 4, 15, 9,71]

Answer:

* + 1. [1, 3, 5] [2, 4, 6]
    2. [-3, 15, 9, 71] [42, 4]

**public** **static** **void** doThis**(**Queue**<**Integer**>** q**)** **{**

Stack**<**Integer**>** s **=** **new** Stack**<>();**

**int** size **=** q**.**size**();**

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

**int** n **=** q**.**remove**();**

**if** **(**n **%** 2 **==** 0**)** **{**

s**.**push**(**n**);**

**}** **else** **{**

q**.**add**(**n**);**

**}**

**}**

System**.*out*.**println**(**q **+** **" "** **+** s**);**

**}**

1. (10 points) What is wrong with the following code? This piece of code incorrectly attempts to find the largest value in a queue of integers. How would you fix it? You will need to write a class with the main(String[] args) {….} method to test the following piece of code, find the problem, and fix it.

**int** largest **=** q**.**remove**(); // assume the first one is the largest.**

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

largest **=** Math**.***max***(**largest**,** q**.**remove**());**

**}**

There are many ways to find the maxi value from a queue list. For this problem, you are not allowed to use any collection utilities; you will have to use the Math.max( largest, q.remove( )) method and you cannot modify the original queue contents. For example, with my queue list (1, 2, 3, 99, -4, 6), the results should look like the following.

Orginal list

[1, 2, 3, 99, -4, 6]

largest = 99

After finding maxi, the list is

[1, 2, 3, 99, -4, 6]

1. (10 points) What is wrong with the following code? This piece of code incorrectly attempts to remove all even values from a stack integers. How would you fix it? You will need to write a class with the main(String[] args) {….} method to test the following piece of code, find the problem, and fix it.

**while** **(!**s1**.**isEmpty**()){**

**int** n **=** s1**.**pop**();**

**if** **(**n **%** 2 **!=** 0**)** **{**

s1**.**push**(**n**);** **// if odd put it back.**

**}**

**}**

1. (20 points) Write a program that find and print the longest string in a stack of strings.

For example, Stack [Hello, Yes, Good Morning, Good Evening, Where is the ATM machine?, no].

The longest string is “Where is the ATM machine?”.

You are not allowed to use collections utilities or transforming it to an array to find the maximum length. You will have to use stack to pop and compare each of the items for the longest string. The original stack must remain intact with the same content after the search operation.

Original Stack:

[Hello, Yes, Good Morning, Good Evening, Where is the ATM machine?, no]

The longest string is: "Where is the ATM machine?"

Stack contents after the search:

[Hello, Yes, Good Morning, Good Evening, Where is the ATM machine?, no]