

CS310 Data Structures Spring 2020

Home Work 2

Points Possible: 20

Name: Sherwin Labadan

RedID: 820229989

- 1) **(2 + 2)** Big-Oh and Run Time Analysis: Describe the worst case running time of the following pseudocode functions in Big-Oh notation in terms of the variable n . Showing your work is not required

```
a) void silly(int n, int x, int y) {  
    if (x < y) {  
        for (int i = 0; i < n; ++i)  
            for (int j = 0; j < n * i; ++j)  
                System.out.println("y = " + y);  
    }  
    else {  
        System.out.println("x = " + x);  
    }  
}
```

n
 $n * n$
 $n * n * n$
 $T(n)$ is $O(n^3)$

```
b) void silly(int n) {  
    for (int i = 0; i < n * n; ++i) {  
        for (int j = 0; j < n; ++j) {  
            for (int k = 0; k < i; ++k)  
                System.out.println("k = " + k);  
            for (int m = 0; m < 100; ++m)  
                System.out.println("m = " + m);  
        }  
    }  
}
```

n
 $n * n$
 $n * n * n$
 $n * n * n$
 $T(n)$ is $O(n^3)$

- 2) **(4 + 4)** Given an **array implementation** of a STACK that holds integers. (Refer to Array Implementation of Stack on Blackboard)
Stack is initialized in the Driver class as follows:

```
StackArray<Integer> intStack = new StackArray<Integer>();
```

- a) Write a new public void member function called `countPosNeg()`

This function counts and displays the number of positive integers and number of negative integers in `intStack`. `intStack` must be returned to its original state after counting.

- i) Write the java code for the function.

```
public void countPosNeg(StackArray<Integer> intStack) {
    StackArray<Integer> temporaryStack = new StackArray<Integer>();
    int x;
    int neg = 0, pos = 0;
    while (!intStack.isEmpty()) {
        x = intStack.pop();
        temporaryStack.push(x);
        if (x < 0) {
            neg++;
        }
        else {
            pos++;
        }
    }
    System.out.println("Negative: " + neg + " \nPositive: " + pos);
    while (!temporaryStack.isEmpty()) {
        x = temporaryStack.pop();
        intStack.push(x);
    }
}
```

- ii) Derive the time function and calculate the BigO to the tightest upperbound of the function `countPosNeg()`

$T(n)$ is $O(n^2)$

- iii) Write a line of code that shows how the `main()` would use this new member function.

`intStack.countPosNeg(intStack);`

- b) Write a **recursive (not iterative)** Java method public Boolean `sameStack(StackArray<E> s2)`

to test whether two stacks contain the same elements. The elements stored in the stack are integers.

One Stack object calls this method with a second stack object, from inside the `main()`
`stack1.sameStack(stack2);`

The function will return true if the stacks contain the same elements and false otherwise.

```

static boolean sameStack(StackArray<Integer> stack, StackArray<Integer> stack2) {
    boolean flag = true;
    while (stack.isEmpty() == false) {
        int x = stack.pop();
        int y = stack2.pop();
        if (x == y) {
            flag = true;
        }
        else {
            flag = false;
            break;
        }
    }
    return flag;
}

```

- 3) (4 + 4) Given an **array implementation** of a CIRCULAR QUEUE that holds integers. (Refer to Array Implementation of Queue on Blackboard)
Using the following input in order:

4 5 67 89 21 3 0 76 34 12

The main function in the Driver Class is as follows:

```

public static void main(String args[])
{
    QueueArray<Integer> numqueue = new
QueueArray<Integer>();

    int i = 0;

    Scanner myObj = new Scanner(System.in);
    do
    {
        int choice = Integer.parseInt(myObj.next());
        if (choice < 35)
            numqueue.enqueue(choice);
        i++;
    }while (i<10);

    do
    {
        System.out.println("Dequeued from the Queue :"+
numqueue.dequeue());
    }while (numqueue.isEmpty()==false);
}

```

a) What is the output of the main function?

```
Dequeued from the Queue :4  
Dequeued from the Queue :5  
Dequeued from the Queue :21  
Dequeued from the Queue :3  
Dequeued from the Queue :0  
Dequeued from the Queue :34  
Dequeued from the Queue :12
```

b) Derive the time function and calculate the BigO to the tightest upperbound of the function
`main()`

$T(n)$ is $O(n^2)$