Claremont McKenna College
Computer Science

CS 62 **Handout 6a** **October 22, 2015**

Sample Old Midterm Exam Sample Solutions

This is an open book/open notes exam, but *nothing electronic* such as a laptop, calculator, etc. is allowed. You have 75 minutes to complete the exam. Answer the problems in the space provided. The exam consists of **seven** problems spread over **eight** pages. The weight of each problem is indicated; the exam will be graded out of 100 points. Give a concise and legible answer to each of the questions.

Read the directions of each problem carefully. If you do not understand a question, please ask for a clarification.

Please write your name, student ID, and email address below.

Problem	Possible	Your Score
1	20	
2	15	
3	15	
4	10	
5	12	
6	20	
7	8	
Total	**100**	

Problem 1 (20 points)

If we were to compile and run the `main` in class `A`, what output would it generate?

```
public class A {

    public int x = 1;

    public static int y = 2;

    public A (int x) {
        this.x = A.y + x;
        y = A.y + x;
    }

    public int m1 (int n) {
        A.y = this.x + y;
        x = n - A.y;
        return y;
    }

    public static int m2 (int x) {
        A.y = y + x;
        return A.y;
    }

    public static void main (String[] args) {

        A a1 = new A(1);
        A a2 = a1;
        a1.m1(2);
        A a3 = new A(3);

        System.out.println(a1.m1(4));  // Answer (a):    5

        System.out.println(a3.m2(5));  // Answer (b):   10

        System.out.println(a2.m1(6));  // Answer (c):    9

        System.out.println(A.m2(7));   // Answer (d):   16
    }
}
```

Problem 2 (15 points)

If we were to compile and run the `main` in class `B`, what output would it generate? If one causes an error, write 'error' as your answer. I am attaching a copy of `Point.java` to the end of this exam.

```
public class B {

    public static void foo (int x) {
        x = x - x;
    }

    public static void foo (int[] x) {
        x[0] = x[0] + x[0];
    }

    public static void bar (Point p) {
        Point[] pa = new Point[] { p, p };
        baz(pa);
    }

    public static void baz (Point[] pa) {
        for (int i = 0; i < pa.length; i++) {
            pa[i] = null;
        }
        pa = null;
    }

    public static void main (String[] args) {
        int[] arr = {11, 22, 33, 44, 55};

        foo(arr[0]);
        System.out.println(arr[0]);        // Answer (a):   11

        foo(arr);
        System.out.println(arr[0]);        // Answer (b):   22

        Point p1 = new Point(10, 20);
        bar(p1);
        System.out.println(p1);            // Answer (c):   10, 20
    }
}
```

Problem 3 (15 points)

Give the running time complexity of each of the following methods in Θ notation. Assume that n is a large enough number. You don't need to justify your answer.

```
// (a)
public static int foo (int n) {
    int sum = 0;
    for (int i = 0; i < n; i = i + 100) {
        sum = sum + sum;
    }
    System.out.println(sum);
    return sum;                          // Answer (a):
}
```

Answer: $\Theta(n)$.

```
// (b)
public static int bar (int n) {
    int sum = 0;
    for (int i = 1; i < n; i = i * 5) {
        sum = sum + 1;
    }
    for (int j = n; j > 0; j = j / 2) {
        sum = sum - 1;
    }
    return sum;                          // Answer (b):
}
```

Answer: $\Theta(log_2 n)$.

```
// (c)
public static int baz (int n) {
    int sum = 0;
    for (int j = 1; j < n; j = j * 10) {
        for (int k = j; k < n; k = k + 4) {
            sum = sum * sum;
        }
    }
    return sum;                          // Answer (c):
}
```

Answer: $\Theta(nlog_{10} n)$.

Problem 4 (10 points)

Consider the following Java method:

```
// Computes the product of the positive elements of 'a' from
// 'from' index to 'to' index of 'a'.
public static int product (int[] a, int from, int to) {
    int p = 1;
    for (int i = from; i <= to; i = i + 1) {
        if (a[i] > 0) {
            p = p * a[i];
        }
    }
    return p;
}
```

Fill in the blanks in the following methods so that `prodRec` computes the same result recursively:

```
public static int prodRec (int[] a, int from, int to) {
    return prodRecAux(a, from, to, 1);
}

private static int prodRecAux (int[] a, int from, int to, int p) {

    if (from > to) {

        _____return p;_____________________________________________;
    }
    if (a[from] > 0) {

        _____return prodRecAux(a, from + 1, to, p * a[from]);_____;
    }

    __________return prodRecAux(a, from + 1, to, p);_______________;
}
```

Problem 5 (12 points)

Determine the running time complexity for the following method using the formal counting technique.

```
public static int p6 (int n) {                    cost     times

    int sum = 0;                                   c0        1

    for (int i = n; i > 0; i = i / 8) {            c1        (log_8 n)

        for (int j = 0; j < n; j = j + 8) {        c2       n(log_8 n)

            sum++;                                 c3       n(log_8 n)

        }

    }

    return sum;                                    c4        1
}
```

To be exact, the count for the first `for` loop would have to be $\lceil log_8 n \rceil + 1$ and the `sum++` line would be counted as $(\lceil log_8 n \rceil + 1)(\lceil n/8 \rceil + 1)$, but I used a bit of approximation. Then, $T(n) = (c_2 + c_3)(nlog_8 n) + c_1(log_8 n) + (c_0 + c_4) = \Theta(nlogn)$.

Problem 6 (20 points)

Consider the following Java program:

```
public class M {
    static boolean even (int i) {
        return ((i % 2) == 0);
    }

    public static int foo (int [] A, int i) {
        if (i < 0) {
            return 0;
        }
        else {
            return A[i] + foo(A, i - 1);  // recursive call
        }
    }

    static int bar (int[] B) {
        if (even(foo(B, B.length - 1))) {
            for (int i = 0; i < B.length; i = i + 1) {
                B[i] = B[i] * B[i];
            }
            return foo(B, B.length - 1);
        }
        else {
            return foo(B, B.length - 1);
        }
    }

    public static void main(String[] args) {
        int[] A = {1, 2, 3};
        int[] B = {1, 2, 4};
        System.out.println(bar(A));
        System.out.println(bar(B));
    }
}
```

(a) What are the two output values generated if we were to run the `main` method?

Answer: 14 followed by 7.

(b) Let `T(n)` be the running time for `foo(A, n-1)`, where `n` is the length of the input argument array `A`. Now, set up a recurrence relation for the function `foo`.

Answer:

$$T(n) = \begin{cases} c + T(n-1) & \text{if } n \geq 0 \\ d & \text{if } n < 0 \end{cases}$$

for some constants c and d.

(c) What is the running time complexity of `foo` in Θ notation?

Answer: $\Theta(n)$.

(d) What is the *worst case* and *best case* running time complexities of `bar` in Θ notation?

Answer: Both are $\Theta(n)$.

Problem 7 (8 points)

We studied merge sort and quick sort in class. We also learned that the running time complexity of quick sort is $\Theta(nlogn)$. Intuitively, where are the n and $logn$ coming from? That is, what in the algorithm contribute to n and what to $logn$?

Answer: n is from the fact that you have to deal with the n elements of the array being compared during partition and $logn$ is coming from the depth of recursive calls with the size of array in each recursive call being roughly equal to half of the size of the current portion of the array being sorted.

```
public class Point {
    public int x;
    public int y;

    public Point () {
        x = 0;
        y = 0;
    }

    public Point (int initX, int initY) {
        x = initX;
        y = initY;
    }

    public String toString () {
        return x + ", " + y;
    }
}
```