CS 173, Fall 2015
Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: Thursday 2 3 4 5 Friday 9 10 11 12 1 2

```
Chop(a_1, ..., a_n; b_1, ..., b_n)  \\ input is 2 lists of n integers, n is a power of 2
        if (n = 1)
                return a_1 b_1
        else
                p = n/2
                rv = Chop(a_1, ..., a_p, b_1, ..., b_p)
                rv = rv + Chop(a_1, ..., a_p, b_{p+1}, ..., b_n)
                rv = rv + Chop(a_{p+1}, ..., a_n, b_{p+1}, ..., b_n)
                rv = rv + Chop(a_{p+1}, ..., a_n, b_1, ..., b_p)
                return rv
```

1. (5 points) Suppose that $T(n)$ is the running time of Chop on an input array of length n. Give a recursive definition of $T(n)$. Assume that dividing the list in half takes $O(n)$ time.

   Solution:

   $T(1) = c$

   $T(n) = 4T(n/2) + dn + f$

2. (4 points) What is the height of the recursion tree for $T(n)$, assuming n is a power of 2?

   Solution: $\log_2 n$

3. (3 points) What is the amount of work (aka sum of the values in the nodes) at level k of this tree?

   Solution: There are 4^k nodes, each containing $f + dn/2^k$. So the total work is $4^k f + 2^k dn$

4. (3 points) How many leaves are in the recursion tree for $T(n)$? (Simplify your answer.)

   Solution: $4^{\log_2 n} = 4^{\log_4 n \log 24} = n^{\log 24} = n^2$

CS 173, Fall 2015
Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: Thursday 2 3 4 5 Friday 9 10 11 12 1 2

```
Crunch(k,n)   \\ inputs are positive integers
        if (n = 1) return k
        else if (n = 2) return k^2
        else
                half = ⌊n/2⌋
                answer = Crunch(k,half)
                answer = answer*answer
                if (n is odd)
                        answer = answer*k
                return answer
```

1. (5 points) Suppose $T(n)$ is the running time of Crunch. Give a recursive definition of $T(n)$.

   Solution: $T(1) = c$, $T(2) = d$

   $T(n) = T(n/2) + f$

2. (4 points) What is the height of the recursion tree for $T(n)$? (Assume that n is a power of 2.)

   Solution: $\log_2 n - 1$

3. (3 points) How many leaves are in the recursion tree for $T(n)$?

   Solution: One.

4. (3 points) What is the big-Theta running time of Crunch?

   Solution: $\Theta(\log n)$

CS 173, Fall 2015
Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: **Thursday** **2** **3** **4** **5** **Friday** **9** **10** **11** **12** **1** **2**

```
01 Process (a_1, ..., a_n: array of integers)
02     if (n = 1)
03         if (a_1 > 8) return true
04         else return false
05     else if (Process(a_1, ..., a_{n-1}) is true and Process(a_2, ..., a_n) is true)
05         return true
06     else return false
```

1. (3 points) If Process returns true, what must be true of the values in the input array?

   Solution: The values in the input array must all be greater than 8.

2. (5 points) Give a recursive definition for $T(n)$, the running time of Process on an input of length n, assuming it takes constant time to set up the recursive calls in line 05.

   Solution:

   $T(1) = c$

   $T(n) = 2T(n-1) + d$

3. (3 points) What is the height of the recursion tree for $T(n)$?

   Solution: n

4. (4 points) What is the big-theta running time of Process?

   Solution: $\Theta(2^n)$

CS 173, Fall 2015
Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: Thursday 2 3 4 5 Friday 9 10 11 12 1 2

```
Twiddle(a_0, ..., a_{n-1}))  \\ input is an array of n integers
        if (n = 2 and a_0 > a_1)
                swap(a_0, a_1)   \\ interchange the values at positions 0 and 1 in the array
        else if (n > 2)
                p = ⌊n/4⌋
                q = ⌊n/2⌋
                r = p + q
                Twiddle(a_0, ..., a_q)   \\ constant time to make smaller array
                Twiddle(a_{q+1}, ..., a_{n-1})   \\ constant time to make smaller array
                Twiddle(a_p, ..., a_r)   \\ constant time to make smaller array
```

1. (5 points) Suppose that $T(n)$ is the running time of Twiddle on an input array of length n. Give a recursive definition of $T(n)$.

   Solution:

   $T(1) = c$, $T(2) = d$

   $T(n) = 3T(n/2) + f$

2. (4 points) What is the height of the recursion tree for $T(n)$, assuming n is a power of 2?

   Solution: $\log_2 n - 1$

3. (3 points) What is the amount of work (aka sum of the values in the nodes) at level k of this tree?

   Solution: $f \cdot 3^k$

4. (3 points) How many leaves are in the recursion tree for $T(n)$? (Simplify your answer.)

   Solution: $3^{\log_2 n - 1} = 1/3(3^{\log_2 n}) = 1/3(3^{\log_3 n \log_2 3}) = 1/3 \cdot n^{\log_2 3}$

CS 173, Fall 2015
Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: **Thursday** **2** **3** **4** **5** **Friday** **9** **10** **11** **12** **1** **2**

01 MyFunc(a_1,..., a_n) \\ input is an array of n integers
02   **for** $i := 1$ to $n - 1$
03         $min := i$
04   **for** $j := i$ to n
05   **if** $a_j < a_{min}$ **then** $min := j$
06   **swap**(a_i, a_{min}) \\ interchange the values at positions i and min in the array

1. (4 points) If the input is $10, 5, 2, 3, 8$, what are the array values after two iterations of the outer loop?

   Solution: After the second iteration, it contains $2, 3, 10, 5, 8$.

2. (4 points) Let $T(n)$ be the number of times that line 5 is executed. Express $T(n)$ using summation notation, directly following the structure of the code.

   Solution: The ith time through the outer loop, the inner loop runs $n - i + 1$ times. So the total number of times that line 5 executes is:

   $$\sum_{i=1}^{n-1}(n - i + 1)$$

3. (4 points) Find an (exact) closed form for $T(n)$. Show your work.

   Solution: If we break apart the sum and then substitute in a new index variable $p = n - i$ we get:

   $$\sum_{i=1}^{n-1}(n-i+1) = (n-1) + \sum_{i=1}^{n-1}(n-i) = (n-1) + \sum_{p=1}^{n-1} p = (n-1) + \frac{n(n-1)}{2}$$

   Simplifying, we get

   $(n-1) + \frac{n(n-1)}{2} = n - 1 + \frac{1}{2}n^2 - \frac{1}{2}n = \frac{1}{2}n^2 + \frac{1}{2}n + 1$

4. (3 points) What is the big-theta running time of MyFunc?

   Solution: $\Theta(n^2)$

CS 173, Fall 2015
Examlet 11, Part A

NETID:

FIRST: | **LAST:**

Discussion: **Thursday** **2 3 4 5** **Friday** **9 10 11 12 1 2**

```
Kitty(a_1, ..., a_n) : list of n positive integers, n ≥ 2)
    if (n = 2) return |a_1 − a_2|
    else
        bestval = 0
        for k = 1 to n
            newval = Kitty(a_1, a_2, ..., a_{k−1}, a_{k+1}, ... a_n)   \\ constant time to remove a_k
            if (newval > bestval) bestval = newval
        return bestval
```

1. (3 points) Describe (in English) what Kitty computes.

   Solution: Kitty computes the largest Kitty difference between two values in the list. Or, equivalently, the largest value minus the smallest value.

2. (5 points) Suppose that $T(n)$ is the running time of Kitty on an input list of length n. Give a recursive definition of $T(n)$.

   Solution:

   $T(2) = c$

   $T(n) = nT(n-1) + d$

3. (3 points) What is the height of the recursion tree for $T(n)$?

   Solution: $n - 2$

4. (4 points) How many leaf nodes are there in the recursion tree for $T(n)$?

   Solution: $\frac{n!}{2}$