ECE368 Mid-Term Exam 1
2019

- **Before the exam:** This is a closed book exam. Calculator, smart devices, and cheatsheets are **NOT** allowed. In choosing your seat, maximize your distance from other students, e.g. seat every other row and/or line.
- **This exam consists of 8 pages** (including this cover sheet). Please check to make sure that all of these pages are present before you begin. Credit will not be awarded for pages that are missing. It is your responsibility to make sure that you have a complete copy of the exam.
- **No questions are allowed** during the exam in order to maintain order and fairness. We only respond to logistics requests. If you have technical doubts that may impact your answers, clearly write down your assumptions as "**ASSUMPTION:..**". We will consider your assumptions in grading.
- **When time is up:**
  - Stop writing immediately
  - Form a line to sign out and turn in the exam
  - Do not chat before you leave the classroom

We strictly enforce these rules. Any violation will be recorded and reported.

"In signing this statement, I hereby certify that the work on this exam is my own and that I have not copied the work of any other student while completing it. I understand that, if I fail to honor this agreement, I will receive a score of ZERO for this exam and will be subject to possible disciplinary action."

`Printed Name:` Sample Copy

`login:`

`Signature:`

DO NOT BEGIN UNTIL INSTRUCTED TO DO SO.

1. (16 points): Give the worst-case time complexity of the following procedures as a function of n using "big O" - `O()` - notation. Clearly justify your answers (answers with no justification will receive zero credit).

1(a). (8 points):

```
void foo (int N) {
    int i = 1, j = 0, total = 0, m, n;
    while (i <= N) {
      i = 2 * i;
      j = j + 1;
    }
    for (m = 1; m <= j; m++)
      for (n = 1; n <= m; n++)
        total *= (n + m);
}
```

$j \leftarrow \lfloor \log_2 N \rfloor + 1$

j

$\sum_{m=1}^{j}\sum_{n=1}^{m} c$

Hint: Consider the following: How many iterations are done in the while() loop? After the while() loop, what are the final value of j ? How many iterations are done in the nested for() loops?

Your choice of the overall big-O complexity (write down one letter): C

(a) O($N * logN$)

(b) O(N^2)

(c) O(($logN$)^2)

(d) O($logN$)

(e) O(N^3)

(f) None above

Your justification:

$$j + \sum_{m=1}^{j}\sum_{n=1}^{m} c = j + c\sum_{m=1}^{j} m = j + c\frac{(1+j)j}{2} = O(j^2)$$

1(b). (8 points):

n

$\sum_{j=i}^{n} 1$

```
void bar (int n) {
  int i = 1, j, x = 0;
  while (i <= n) {
    if (i >= (n / 2)) {
      for (j = i; j <= n; j++)
        x = x * 2;
    }
    i = i + 1;
  }
}
```

Hint: Consider the following: will the values of x affect the program execution?

Your choice of the overall big-O complexity (write down one letter): C

(a) O($n * logn$)

(b) O(n)

(c) O(n^2)

(d) O($i * j$)

(e) O(n^3)

(f) None above

Your justification:

$$C + \sum_{i=\frac{n}{2}}^{n}\sum_{j=i}^{n} d + \sum_{i=1}^{n/2} e = C + d\sum_{i=\frac{n}{2}}^{n}(n-i+1) + e\frac{n}{2} = c + e\frac{n}{2} + d\left(n\cdot\left(\frac{n}{2}+1\right) - \frac{\left(\frac{n}{2}+n\right)\left(\frac{n}{2}+1\right)}{2} + \frac{n}{2}+1\right)$$

$$= O(n^2)$$

2. (14 points): Consider the following sorting algorithm: You are given an unsorted array ($A[\,]$) of n elements. Divide the n elements of the unsorted array into $\sqrt{n}$ groups of $\sqrt{n}$ elements each (for simplicity, assume that $\sqrt{n}$ is an integer). Find the largest element of each group and insert it into an auxiliary array ($B[\,]$). Find the largest of the elements in this auxiliary array. This will be the largest element in the array $A[\,]$. Insert it into an output array ($C[\,]$) in the correct position. Then replace this element in the auxiliary array by the next largest element of the group from which it came. Again find the largest element of the auxiliary array. This is the second largest element of the entire array. Repeat the process until the entire array has been sorted.

2(a). (8 points): Apply the above sort to `[11, 21, 39, 48, 43, 3, 33, 36, 20]`.

- Show how the unsorted array will be divided into groups:

(11, 21, 39), (48, 43, 3), (33, 36, 20)

X X X   X X X   X X X

In particular, every time you insert any element into **the auxillary array**, show the contents of the auxiliary array immediately after the insertion.

- Auxiliary array **before** 1st insertion: 39, 48, 36
- Auxiliary array after 1st insertion: 39, 43, 36
- Auxiliary array after 2nd insertion: 39, 3, 36
- Auxiliary array after 3rd insertion: 21, 3, 36
- Auxiliary array after 4th insertion: 21, 3, 33
- Auxiliary array after 5th insertion: 21, 3, 20
- Auxiliary array after 6th insertion: 11, 3, 20
- Auxiliary array after 7th insertion: 11, 3, -inf   or   11, 3
- Auxiliary array after 8th insertion: -inf, 3, -inf   3
- Auxiliary array after 9th insertion: -inf, -inf, -inf   empty

2(b). (4 points): What is the time complexity of the above sorting algorithm?

Your choice of the big-O complexity (write down one letter): C

(a) $O(n)$

(b) $O(n * logn)$

[✓ (c) $O(n * \sqrt{n})$]

(d) $O(n^2)$

(e) $O(n^3)$

(f) None above

Your justification:

$$a\sqrt{n}\sum_{i=1}^{\sqrt{n}} i + bn\sqrt{n}$$

2(c). (2 points): What is the space complexity of the preceding sorting algorithm (in `O()` notation)? Justify your answer.

Your choice of the big-O complexity (write down one letter): a

(a) $O(n)$

(b) $O(n * logn)$

(c) $O(n * \sqrt{n})$

(d) $O(n^2)$

(e) $O(n^3)$

(f) None above

Your justification:

$$\left.\begin{array}{l} \text{len}(C) \propto n \\ \text{len}(B) \propto \sqrt{n} \end{array}\right\} O(n)$$

3. (10 points): Arrays and linked lists.

You are asked to implement a list data structure using either^{p} a statically-defined array or a singly-linked list. Assume that the size of a data element is X times the size of a pointer. The size of the statically-defined array in an array implementation is N (*i.e.,* the space for the array has to be allocated altogether, and it can hold a maximum of N data elements), and the current number of data elements in the list is C.

3(a). (4 points): If you are given that X = 4, N = 200, and C = 60, which data structure (singly-linked list or statically-defined array) uses less space?

Your choice (write down one letter): a

(a) singly-linked list

(b) statically-defined array

(c) None above

Explain your answer:

statically-defined array : $4p \times 200$

SLL : $(4+1)p \times 60$

3(b). (6 points): Derive an expression (in terms of X, N, and C) that determines when the singly-linked list implementation uses less space than the array implementation? Explain your answer. Assume that the array implementation uses implicit addressing (i.e., you do not need to store the next index of an element).

$$(x+1)p \cdot C < Xp \cdot N$$

$$(X+1)C < XN$$

$$\frac{C}{N} < \frac{X}{X+1}$$

25a + 10b + 5c + d = 30

4. (10 points): Given an infinite number of quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent), we need to calculate *k*, the number of ways of representing *n* cents.

(1)+(6) + (1+3+5) + 2

(a). (4 points): When $n = 30$, write down the value of *k* here: 18

(b). (6 points): The following recursive function calculates *k*.

(5,1)

5 ~ 2
10 ~ 3
15 ~ 4
20 ~ 5

```
 1 int makeChange(int n, int denom) {
 2
 3     int next_denom = 0;
 4     int k = 0;
 5
 6     switch (denom) {
 7     case 25:
 8       next_denom = 10;
 9       break;
10     case 10:
11       next_denom = 5;
12       break;
13     case 5:
14       next_denom = 1;
15       break;
16     case 1:
17       return 1;
18     }
19
20     for (int i = 0; i * denom <= n; i++)
21       k += makeChange(n-i*denom, next_denom);
22     }
23     return k;
24 }
```

In line 21, the two arguments of the recursive call are missing. Fill in the blanks.

For grading purposes only. Do not write on this page.

Question	Credits
1	
2	
3	
4	
Total	