CST 370 Design and Analysis of Algorithms
Spring A 2020
Midterm-I

Name: ___Ivan Alejandre_________________________________

Four-digits Class ID: ___8889_____________________________

- Test time is **2 hours and 30 minutes**.
- Note that there are **12 problems**
- This is a **closed book exam**. You **can't use a calculator** during the exam. However, as a reference during the exam, you can prepare "two pages (= total of 4 sides)" **cheat sheet**. The cheat sheet can be typed or hand-written.
- If possible, enter your answers directly into the Word file to increase the readability of your answers. However, if it is too difficult or time consuming to type in a Word file, please write down your answer on paper. Then, take a picture and insert the picture into the Word file.
- During the exam, you must **sign into the Zoom session** and **turn on the video**. We will record the video. However, **turn off the audio** on your computer.
- If you have a question during the exam, please use "Chat" in Zoom. I will answer.
- When you finish the exam, submit your file in PDF format (optional Word format) on the iLearn.
- Use your time wisely—make sure to answer the questions you know first.
- Read the questions carefully.

1. (5 points) Here is the selection sort algorithm as in the textbook.

Suppose that you have an array of length *N* consisting of alternating 1's and 2's, starting with 1. For example, below is an array for *N* =16:

1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2

How many compares does selection sort make to sort the array as a function of N?

C(n) = SUM[i=0 → n-2] [(n – 1) - (i + 1) + 1] = (n – 1 – i)

2. (5 points) Determine what is calculated by the following recursive function:

```
unsigned f(int n)
{
        if (n < 0)
                return f(-n);
        if (n < 10)
                return n;
        return f(n/10);
}
```

Given any n, this function returns a number between 0 and 9. The first if statement takes any negative n and passes it into the function again as its absolute value. The second if block returns our solution if it is less than 10. The third block calls the function again passing in n divided by 10 (which is a int division). It'll continually call itself until n is in the range of 0-9 inclusive.

3. (10 points) Write a recursive function (in Pseudocode) that returns the number of digits in a nonnegative integer (decimal).

```
digits(int n)
// returns the number of digits in a given positive integer
// digits is global
digits ← 1;
Algorithm digits(n)
1. if (n < 10)
2.         return digits;
3. else
```

```
4.    digits ← digits + 1;
5.    digits(n/10);
```

4. (5 points) Rank the following in order of increasing complexity:

$O(N)$, $O(N*\log\log N)$, $O(2/N)$, $O(N*\log(N^2))$, $O(2^N)$.

$O(N)$
$O(2/N)$
$O(N*\log\log N)$
$O(N*\log(N^2))$
$O(2N)$

5. (10 points) Based on the definitions of O, Θ, and Ω, determine whether the following assertions are true or false.

(a) $n * n * (n + 1) + 7 * n * n \in O(n^3)$ ([✓ true]/false)

(b) $2n^3 + 7n + 5 \in \Theta(n^2)$ (true/[✓ false])

6. (5 points) Solve the following recurrence relation using the **backward substitution** as we discussed in the class. In the problem, you should present the **intermediate steps** and **final time complexity** of the recurrence relation.

$F(n) = F(n – 1) + 5$, for $n > 1$ // recurrence relation

$F(1) = 0$ // initial condition

$F(n) = F(n – 1) + 5$ // replace $F(n – 1)$ w/ $F(n – 2) + 5$
$= [F(n – 2) + 5] + 5$
$= F(n – 2) + 5 + 5$ // replace $F(n-2)$ w/ $F(n-3) + 5$
$= [F(n – 3) + 5] + 5 + 5$
$= F(n-3) + 5 + 5 + 5$
…
$F(n-k) + 5*k$ // $k = n – 1$
$F(n- (n - 1) + 5*(n – 1)$
$F(1) + 5n – 5$
$0 + 5n – 5$
$n \rightarrow O(N)$

7. (10 points) Consider a sorting program in C++.

```
void my_sorting(int arr[], int n)
{
   int i, j, k, temp;
```

```
 4.    i = 0;
 5.    while (i < n-1)
 6.    {
 7.        k = i;
 8.        j = i+1;
 9.        while (j < n)
10.        {
11.            if (arr[j] < arr[k])
12.            {
13.                k = j;
14.            }
15.            j = j+1;
16.        }
17.        temp = arr[k];
18.        arr[k] = arr[i];
19.        arr[i] = temp;
20.        i = i+1;
21.    }
22. }
```

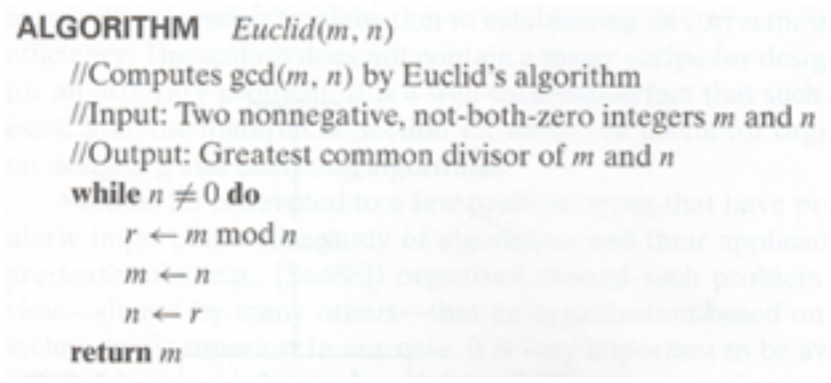
(a) Present the **basic operation** of the program. In your answer, you should **also present the line number** of the basic operation clearly.

Line 9, the < comparator is the basic operation. It is executed n-1 times in the inner loop which is executed n-1 times.

(b) Present the **time complexity** of the algorithm in the **O notation**.

$O(n^2)$

8. (10 point) This is the Euclid's algorithm to find the greatest common divisor (gcd).



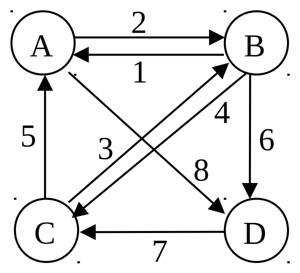
```
ALGORITHM  Euclid(m, n)
    //Computes gcd(m, n) by Euclid's algorithm
    //Input: Two nonnegative, not-both-zero integers m and n
    //Output: Greatest common divisor of m and n
    while $n \neq 0$ do
        $r \leftarrow m$ mod $n$
        $m \leftarrow n$
        $n \leftarrow r$
    return m
```

Determine the return value for the ***Euclid (60, 25)*** using the algorithm. Show the intermediate steps.

Gcd(60, 25) = gcd(25, 60 % 25) → gcd(25, 10)
gcd(25, 10) = gcd(10, 25 % 10) → gcd(10, 5)
gcd(10, 5) = gcd(5, 10 % 5) → **gcd(5, 0)**

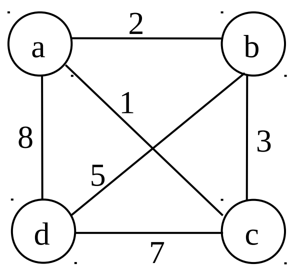
gcd = 5

9. (5 points) Represent the following weighted digraph in **an adjacency list** as we covered in the class.



A → B → D
B → A → C → D
C → A → B
D → C

10. (15 points) Assume that you want to solve the Traveling Salesman Problem for a graph given below. Your starting vertex is the vertex (a).



(a) Present a travelling path to solve the problem.
A → C → B → D → A

(b) Based on the travelling path of your answer to the question (a), present the travelling distance (or cost) of the path.
(17)

(c) Do you have only one optimal solution for the graph? (Yes/No). If your answer is no, present other optional path(s).
A → C → D → B → A (15)

11. (10 points) Suppose you have three jars, A, B, and C, in a room. Jar A has 5 large black balls, 4 large red balls, and 3 large green balls. Jar B has 5 small black balls, 4 small red balls, and 3 small green balls. Jar C is empty. Now, you will pick a few balls from the jar A in the dark and place them in the jar C. After that, you will pick a few balls from the jar B in the dark and place them in the jar C. Note that the color of the selected balls at the jars A and B cannot be confirmed because the surroundings are dark. Also, the numbers of balls selected from the jars A and B need not always be the same. Once you're done, you can turn on the lights in the room and see the balls in the jar C.

(a) Assuming the **best case occurs**, what is the minimum number of balls you have to choose to **get a matching pair**? Here, a matching pair means that there must be one large ball and one small ball of the same color in the jar C. But the **color** itself of the pair **is not important**. **Present just the number of balls**. You **don't need to explain your answer**.

2 balls, lucky draw

(b) Assuming the **worst case occurs**, what is the minimum number of balls you have to choose to get a matching pair? Here, a matching pair means that there must be one large ball and one small ball of the same color in the jar C. But the **color** itself of the pair **is not important**. **Present just the number of balls**. You **don't need to explain your answer**.

12 balls, 11 from A, 1 from B.

12. (10 points) Consider the following algorithm.

```
// Assume that n is a positive integer (= i.e. n > 1), and A[1..n] is a global array.
// Note that the index of array A starts from one, not zero.
// And also, don't forget the array A in the algorithm is global.
Algorithm DoSomething (n)
1.  if (n = 1)
2.      print the current content of the whole array A on the screen;
3.  else
4.      for i ← 1 to n do
5.          DoSomething(n – 1);    // Recursive call.
6.          if n is odd
7.              swap A[1] and A[n];
8.          else
9.              swap A[i] and A[n];
10. return;
```

(a) Present execution result of the algorithm where an array **A** has "**5**"and *n* is 1.
[5]

(b) Present execution result of the algorithm where an array **A** has "**5, 7**"and *n* is 2.
[5, 7]
[7, 5]

(c) Present execution result of the algorithm where an array A has "**5, 7, 9**"and *n* is 3.
[5, 7, 9]
[7, 5, 9]
[9, 5, 7]