Question 1

1 / 1 pts

Assume that you want to solve a **knapsack problem**. Your knapsack capacity is "**8**" and there are four items as below. Note that **each item has only one** quantity. To bring the maximum value, **which item(s) will you bring to the knapsack**?

Item	Capacity	Value
A	1	$15
B	5	$10
C	3	$9
D	4	$5

Your Answer:

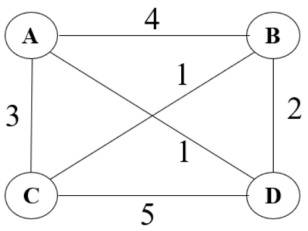
Items; A,C,D

Answer is "A", "C", and "D" because their value is $29 which is the biggest value among all combinations.

Question 2

1 / 1 pts

Present **the optimal cost** of the following TSP (= Traveling Salesman Problem). For the problem, you have to start from the vertex **A**.



- none of these
- 8
- 12
- 7

Question 3

1 / 1 pts

Select the main purpose of the following pseudocode.

```
1. Algorithm DoWhat (A[0..n-1])
2. v1 ← A[0]
3. i ← 1
4. while ( i < n ) do
5.     v1 ← v1 + A[i]
6.     i ← i + 1
7. v2 ← v1/n
8. return v2
```

- Find a max.
- None of these.
- Find a sum.
- Find a min.
- **Find an average.**

Question 4

0.5 / 0.5 pts

Select the execution result of the following code fragment.

```
1. list<int> myList = {30, 50, 70};
2. list<int>::iterator ptr = myList.begin();
3. ptr++;
4. ptr++;
5. cout << *ptr << endl;
```

- 50
- 30
- **70**
- None of these.

Question 5

1 / 1 pts

Present the execution result of the following code fragment.

```
1. queue<int> q;
2. q.push(100);
3. q.push(200);
4. q.push(150);
5. q.pop();
6. cout << q.front() << endl;
7. q.pop();
8. q.push(100);
9. cout << "Size: " << q.size() << endl;
```

Your Answer:

200
Size; 2

Answer:

```
200
Size: 2
```

Question 6

1 / 1 pts

Assume that there is an **undirected graph G with ten vertices**. What's the maximum possible number of edges in the graph? Write your answer in a numerical number such as 0, 1, 2, 3, etc.

45

ers

45

Question 7

0.5 / 0.5 pts

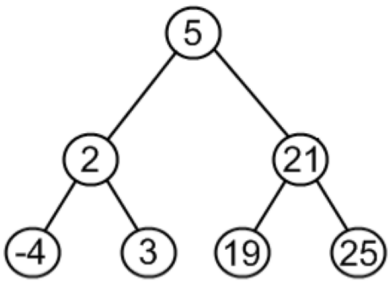
If you have a **sparse graph**, _____ is a better representation in terms of memory usage. Select the correct answer.

- adjacency matrix
- adjacency tree
- adjacency stack
- adjacency list

Question 8

0.5 / 0.5 pts

This is a **binary search tree**.

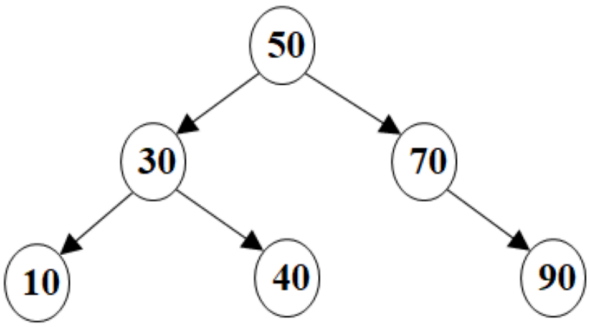


- (!) True
- False

Question 9

0 / 0.5 pts

Based on the definition of our lecture, this is **a graph**.



- (er) True
- (ed) False

Question 10

1 / 3 pts

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. Thus, there are **total 24 balls**. Now, you will pick one or more balls from the jar A in the dark and place them in the jar C. After that, you will pick one or more 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 can not 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.

There are **three questions** in the problem. Write your answer to each question (a), (b), and (c) clearly.

(a) 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.**

(b) Assuming the **best case occurs**, what is the minimum number of balls you have to choose to **get three matching pairs of each color (= black, red, green)**? In other words, you should have one pair of large and small black balls, one pair of large and small red balls, and one pair of large and small green balls. **Present just the number of balls. You don't need to explain your answer.**

(c) Assuming the **worst case occurs**, what is the minimum number of balls you have to choose to **get three matching pairs of each color (= black, red, green)**? In other words, you should have one pair of large and small black balls, one pair of large and small red balls, and one pair of large and small green balls. **Present just the number of balls. You don't need to explain your answer.**

Answer: (a) You have to pick "11" balls to have a matching pair in the worst case. You need to select 10 balls from the jar A to guarantee that the balls with different colors should be included. In the worst case, you could choose 5 black balls, 4 red balls, and 1 green ball. For a matching pair, you have to choose one ball from the jar B. No matter which ball you select from the jar B, there will be one matching pair.

(b) You need to pick only "6" balls to have three matching pairs in the best case. You select 3 balls from the jar A with different colors and 3 balls from the jar B with different colors. This is the "best case" situation.

(c) You have to pick "20" balls to have three matching pair in the worst case. You need to select 10 balls from the jar A to guarantee that the balls with different colors can be included. In the worst case, you could choose 5 black balls, 4 red balls, and 1 green ball. Similarly, you need to select 10 balls from the jar B to guarantee that the balls with different colors can be included. Thus, you have to choose total 20 balls to have three matching pairs with three different colors.

Question 11

1 / 1 pts

Read the following pseudocode carefully and select the correct answer to **Q1** and **Q2**.

```
Algorithm DoSomething (A[0..n-1])
1. value ← A[0]
2. i ← 1
3. while ( i < n ) do
4.     if (A[i] > value)
5.         value ← A[i]
6.     i ← i + 1
7. return value
```

(**Q1**) The **< operation** in the **line number 3** can be a **basic operation**.

(**Q2**) Present the **time complexity** of the algorithm using the theta(= **Θ**) notation.

Q1 Yes

Q2 theta(n)

Question 12

1 / 2 pts

Read the following algorithm carefully and then write your answers to the **two questions** below.

```
// In the algorithm, assume that there are two arrays named
B[0..n-1]
// and C[0..n-1] in addition to the input array A[0..n-1].
Algorithm DoSomething (A[0..n-1])
1.  i <- 0
2.  while (i < n) do
3.      B[i] <- 0
4.      i <- i + 1
5.
6.  i <- 0
7.  while (i < n-1) do
8.     j <- i + 1
9.     while (j < n) do
10.         if A[i] > A[j]
11.             B[j] <- B[j] + 1
12.         else
13.             B[i] <- B[i] + 1
14.         j <- j + 1
15.     i <- i + 1
16.
17. i <- 0
18. while (i < n) do
19.     C[B[i]] <- A[i];
20.     i <- i + 1
21.
22. return;
```

(a) Present the **basic operation** of the algorithm. When you describe the basic operation, you should also present **the line number** of the basic operation clearly.

(b) Present the **time complexity** of the algorithm using the **theta (= Θ) notation**. If it's difficult for you to write the symbol Θ for your answer, just use the text "theta".

(a) Line number 9 (<), 10 (>), 14(+) can be a basic operation.

(b) Time complexity is theta(n^2).

Question 13

1 / 1 pts

Assume that your program **calculates the average of all numbers in an array**. The time efficiencies of best case and worst case of the program are the same.

ct!

- (•) True
- ( ) False

Question 14

0.5 / 0.5 pts

Assume that Dr. Byun assigned a programming project which requires the time complexity of **$O(n^2)$**. If your program's basic operation runs **(2*n*logn + 5)** times, you can say that your program meets the project requirement.

ct!

- (•) True
- ( ) False

Question 15

2 / 2 pts

Based on the definitions of **O**, **Θ**, and **Ω**, determine whether the following assertions are true or false.

(a) $2n^2 + 5n + 1 \in \Theta(n^2)$

(b) $7 * n * n * (n + 1) \in \Omega(n^2)$

(c) $2 * n * (n - 1) \in O(n*\log n)$

(d) $4 * (n * n) + 3 * n \in \Omega(n^2)$

(a) True

(b) True

(c) False

(d) True

Question 16

1 / 1 pts

Read the following pseudocode carefully and answer to the following questions.

```
ALGORITHM DoSomething (A[0..n - 1])
1. for i ←0 to n - 2 do
2.     for j ←i + 1 to n - 1 do
3.         if A[i] == A[j]
4.             return false
5. return true
```

(a) Time complexity of the algorithm is **$\Theta(n^2)$**.

(b) Time complexity of the algorithm is **$O(n^2)$**.

(a) False

(b) True

Question 17

1 / 1 pts

Consider the following recurrence relation and initial condition for a recursive algorithm

M(n) = 2 * M(n-1) + 5 // recurrence relation

M(1) = 5 // initial condition

This is the first step of the backward substitution to get the time complexity of the algorithm. Select the correct result of the first substitution.

M(n) = 2* M(n-1) + 5
= ?

- 2^2 * M(n-2) + 5 + 5
- 2 * M(n-2) + 5 + 5
- 2 * M(n-2) + 2 * 5 + 5
- None of these.
- ✓ 2^2 * M(n-2) + 2 * 5 + 5

Question 18

1 / 1 pts

Consider the following recursive algorithm.

```
1. Algorithm F(n)
2. if n = 1
3.    return 1
4. else
5.    return F(n - 1) + 2 * n - 1
```

Assume that **C(n)** indicates the **number of comparison operation(s)** to be executed in the **line number 2**. Using the notation, you are going to represent the **initial condition** of the recursive algorithm. Select the correct one.

- ✓ C(1) = 1
- C(0) = 1
- C(1) = 0
- None of these.
- C(0) = 0

Question 19

1 / 1 pts

Consider the following recursive algorithm.

```
1. Algorithm F(n)
2. if n = 1
3.     return 1
4. else
5.     return F(n - 1) + 2 * n - 1
```

Assume that **C(n)** indicates the **number of comparison operation(s)** to be executed in the **line number 2**. Using the notation, you are going to represent the **recurrence relation** of the recursive algorithm. Select the correct one.

- C(n) = n-1
- C(n) = C(n-1) + 1
- C(n) = n
- None of these.
- C(n) = C(n-1) + 2

Question 20

2 / 2 pts

Consider the following recursive function:

```
Algorithm Q(n)
1. if n = 1
2.     return 1
3. else
4.     return Q(n - 1) + 2 * n - 1
```

(a) Return value for **Q(1)**.

(b) Return value for **Q(2)**.

(c) Return value for **Q(3)**.

- (a) 1
- (b) 4
- (c) 9

Question 21

0.5 / 0.5 pts

Consider the **brute-force string-matching algorithm** covered in the class. Present **an example of a text with 7 characters** and **an example of a pattern with 3 characters** that constitutes the **worst-case** for the brute-force string-matching algorithm.

Your Answer:

AAAAAAB
AAB

Example Text: AAAAAAA

Example Pattern: AAB

After 3 character comparisons, the algorithm should shift one character, which requires a lot of character comparison (= worst case).

Question 22

0 / 1 pts

Let A[0..n-1] be an array of *n* integer numbers. Assume that all numbers in the array are distinct. In the array, a pair of two numbers (A[i], A[j]) is called an **inversion** if i < j and A[i] > A[j]. For example, there is one inversion in the array A with the values 5, 15, and 10. Note that the index 1 has 15 and the index 2 has 10.

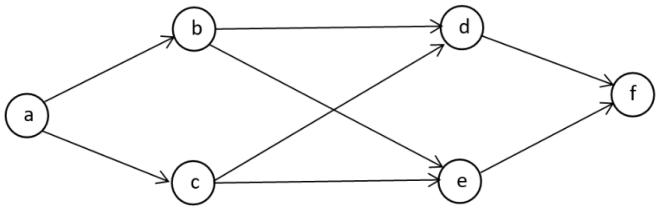
For an array A with the 3 values, what is the **largest number of inversions possible** in the array? Additionally, present **a sample array** with 3 values.

The largest number of inversions for the array with 3 values is **3**. For example, if we have an array A with the values 30, 20, and 10, then there are three inversions of (30, 20), (30,10), and (20,10).

Question 23

2 / 2 pts

Assume that you are going to conduct **DFS (= Depth-First Search)** for the following graph from the **vertex a**. Select the visiting sequence from the number 1 to fill the mark array as you learned in the lecture. When you traverse the graph, you should follow our convention (= alphabetical order of vertex characters).

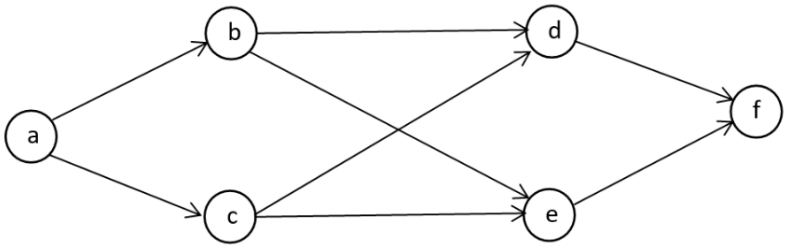


a	1
b	2
c	6
d	3
e	5
f	4

Question 24

2 / 2 pts

Assume that you are going to conduct **BFS (= Breadth-First Search)** for the following graph from the **vertex a**. Select the visiting sequence from the number 1 to fill the mark array as you learned in the lecture. When you traverse the graph, you should follow our convention (= alphabetical order of vertex characters).



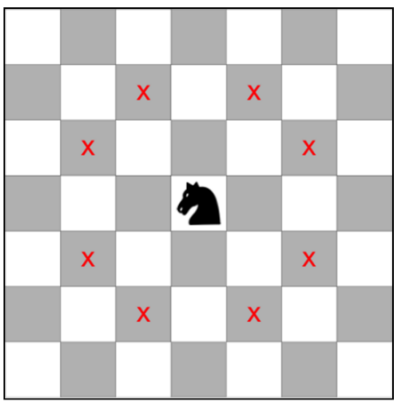
a	1
b	2
c	3
d	4
e	5
f	6

Question 25

1 / 2 pts

[Puzzle] What is the **minimum number of moves** needed for a **chess knight** to go **from one corner of a 100 × 100 board to the diagonally opposite corner**? For this problem, you **don't need to explain your answer**. **Write just the minimum number** of moves.

Note that the knight's moves are L-shaped: It can move two squares horizontally and one square vertically, or two squares vertically and one square horizontally. For example, let's assume that a chess knight is in the middle square of the following board. Then, it can move to 8 different squares as the diagram indicates.



Answer: **66 moves**.

Let's say that you start from the lower left corner square which corresponds to row 1 and column 1. Then, the goal is to reach to the square at the row 100 and column 100.

To be there in the minimum moves, you should go through **33 different squares** such as **(1, 1) --> (4, 4) --> (7, 7) --> ... --> (100, 100)**.

From one square (for example (1, 1)) to the next square (for example (4,4) in the sequence, you need two moves.

Thus, you need total 66 moves (= 2 X 33).