Anonymous code: _____

	1	2	3	4	5	6	7	8	Grade
Results:									

Basic question 1: Insertion sort

Assume you want to sort each of the following arrays using insertion sort:

A. `[4, 7, 2, 2, 5, 4]`
B. `[2, 2, 4, 4, 5, 5]`
C. `[1, 7, 2, 6, 3, 4]`
D. `[7, 6, 5, 4, 2, 1]`
E. `[6, 3, 5, 3, 6, 1]`

Which array is the worst (uses the most number of array accesses),
and which is the best (uses the least)?

Worst: _____

Best: _____

Brief explanation:

Basic question 2: Linearising a BST into a list

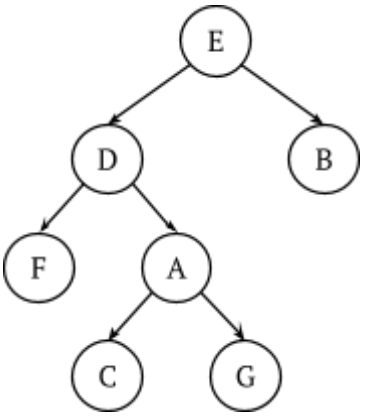
Assume the following code that transforms a binary search tree into a list of its values:

```
def f(bst) -> list:
    result = empty list
    agenda = empty stack or queue
    add the root of bst to agenda

    while agenda is not empty:
        node = remove from agenda
        if node is not null/None:
            add node.left to agenda
            append node.value to the end of result
            add node.right to agenda

    return result
```

Now assume you give it the BST to the right as input:



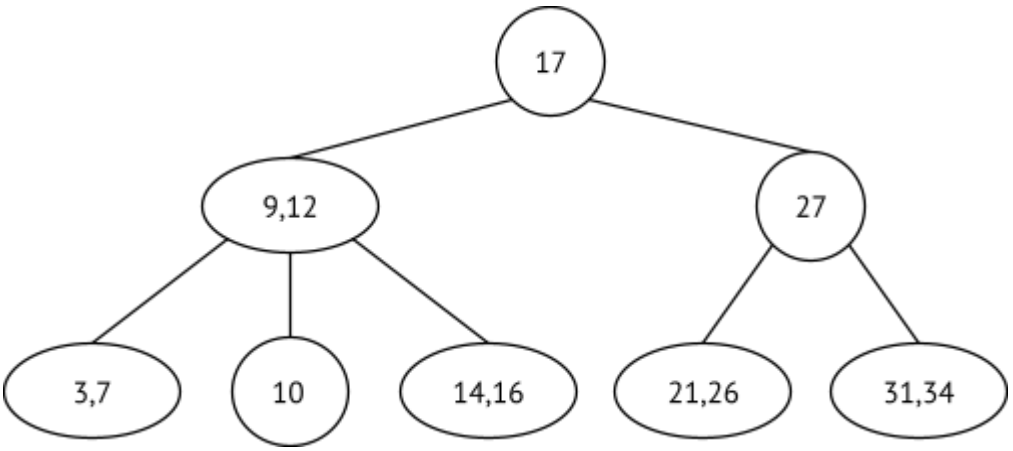
In what order will the values be returned…

(A) …if we use a ***stack*** as the agenda?

(B) …if we use a ***queue*** as the agenda?

Basic question 3: Insertion in 2-3 trees

Given the following 2-3 tree:



Insert first 32 and then 25 into this 2-3 tree. Show how the tree looks after each insertion.

After inserting 32:

After inserting 25:

Basic question 4: Priority queues

Assume the following binary min-heap:

0	1	2	3	4	5	6	7	8	9	10	11
2	7	5	10	13	8	16	16	13	15		

Insert the number 6 into the heap and show the result here:

0	1	2	3	4	5	6	7	8	9	10	11

Delete the minimum element from the resulting heap and show the result here:

0	1	2	3	4	5	6	7	8	9	10	11

Optionally you can also draw the heaps as trees here:

Basic question 5: Hash tables

Here is an open addressing hash table of strings, which uses standard modular hashing and +1 linear probing:

0	1	2	3	4	5	6	7	8	9
H	A	V	E				F	U	N

The hash codes for each of the strings are the following:

$$h(\text{A}) = 38,\ h(\text{E}) = 20,\ h(\text{F}) = 97,\ h(\text{H}) = 49,\ h(\text{N}) = 27,\ h(\text{U}) = 57,\ h(\text{V}) = 42$$

In which order can the elements have been inserted into the hash table?

Check the box(es) ☑ with the correct alternative(s) – there can be more than one:

- ☐ V F U N H A E
- ☐ H A V E F U N
- ☐ F U N H A V E
- ☐ F V U N H E A
- ☐ F V N U H A E

Now, pick one of the correct alternatives and insert the elements in ***reversed order*** into the following linear probing hash table:

0	1	2	3	4	5	6	7	8	9

In which order did you insert the elements? _____

Basic question 6: Quicksort partitioning

Perform a quicksort partitioning in-place of the following array:

0	1	2	3	4	5	6	7	8
17	97	21	34	37	1	13	42	33

Use the element at index 3 (i.e., the number 34) as pivot, and then partition. Recall that the partitioning algorithm also swaps the pivot into the correct place afterwards.

Note: quicksorting the left and right parts of the partition is not part of this question.

Write down how the array looks after the partitioning:

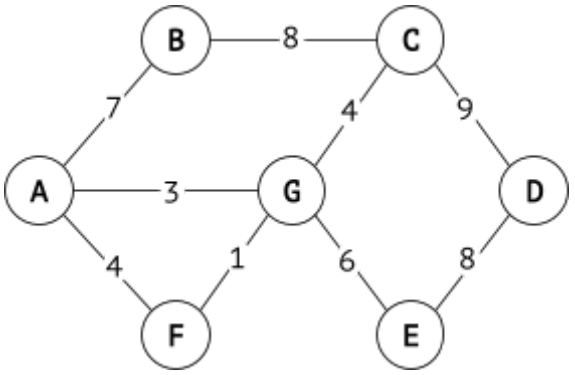
0	1	2	3	4	5	6	7	8

What sequence of swaps did you make when partitioning the array?

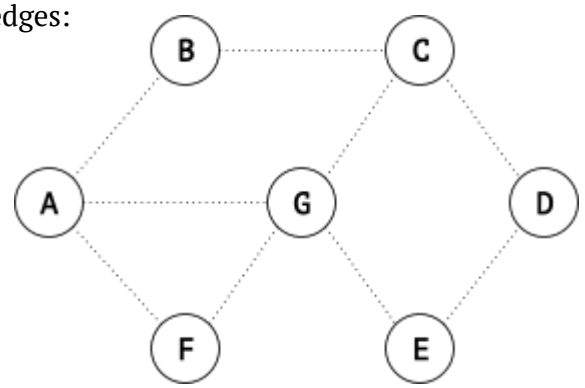
If you used a different algorithm from that of the course, explain it here:

Basic question 7: Graphs

Here is a weighted undirected graph:



Perform **Prim's algorithm** to construct the minimum spanning tree (MST) for the graph.
Draw the MST by filling the dotted edges:



List the edges of the MST *in the order they are produced* by the algorithm.
Do this for two different starting vertices.

- Write the edges in the form AC, DF, ...
- You do not have to use all rows.

Starting vertex: A

1	
2	
3	
4	
5	
6	
7	
8	

Starting vertex: D

1	
2	
3	
4	
5	
6	
7	
8	

Basic question 8: Complexity

Here is a sorting algorithm that uses a minimum priority queue for sorting an array of numbers.

```
def sort(array):
    S = new empty binary heap

    for each x in array:
        insert x into S

    i = 0
    while S is not empty:
        y = delete minimum from S
        array[i] = y
        i += 1
```

What is the worst-case asymptotic time complexity in the size *n* of `array`? Answer for each of the two loops, and the total complexity. Write your answer in O-notation, and be as exact and simple as possible.

Complexity of the for loop: O(_____)

Complexity of the while loop: O(_____)

Complexity of the function sort: O(_____)

Brief explanation:
(Alternatively, you can add comments directly to the code above if you want.)