Q1 Academic Honesty
1 Point

It is a violation of the Academic Integrity Code to look at any reference material other than your textbook and lecture notes, or to give inappropriate help to someone or to receive unauthorized aid by someone in person or electronically via messaging apps such as WhatsApp. Academic Integrity is expected of all students of Hacettepe University at all times, whether in the presence or absence of members of the faculty. Do NOT sign nor take this exam if you do not agree with the honor code.

Understanding this, I declare I shall not give, use or receive unauthorized aid in this examination.

Signature (Specify your name and surname as your signature)

Mehmet Taha USTA MTUSTA

While answering the following questions, please consider the implementations that we discussed in our lectures unless stated otherwise.

Q2 Heaps
16 Points

A 4-heap is an array representation of a complete quaternary (4-ary) tree in which the key in each node is greater than (or equal to) the keys in each of its children.

Q2.1
4 Points

Consider that a key has the index k. How you can define the indices of its four (potential) children as a function of k?
While answering this question, please assume 1-based indexing (the initial element is assigned the index 1).

4k + 1

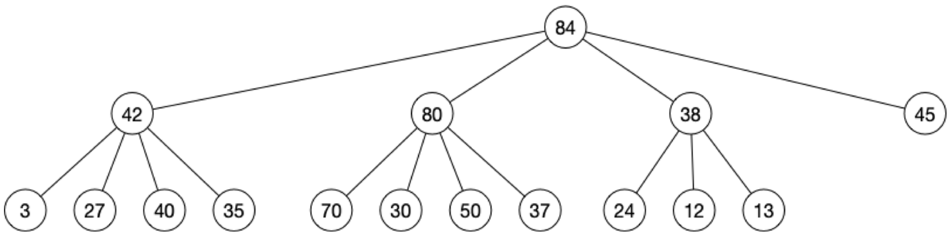
Q2.2
4 Points

What is the maximum number of comparisons for a delmax (delete-the-maximum) operation on a 4-heap as a function of the number of keys N?
While answering this question, please use ~–notation (tilde notation).

At most 2logn compares
~(2logn)

Q2.3
4 Points

Consider the 4-heap given below



whose level-order traversal is as follows:

```
84 42 80 38 45 3 27 40 35 70 30 50 37 24 12 13
```

Perform a delmax operation on this 4-heap, and provide the resulting 4-heap in level order below.

80 42 70 38 45 3 27 40 35 13 30 50 37
24 12

Q2.4
4 Points

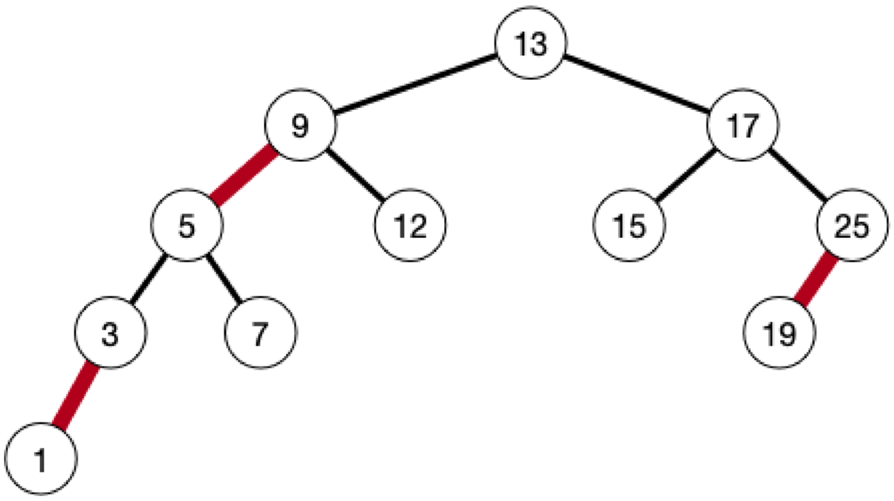
Re-insert 84 to the 4-heap you computed above in Question 1.3, and provide the resulting 4-heap in level order below.

84 42 70 80 45 3 27 40 35 13 30 50 37 24 12 38

Q3 Red-black trees

12 Points

Consider the following left-leaning red-black (LLRB) BST.



Suppose that you insert the following keys into the LLRB above in the given order.

For each insertion, (i) give the number of color flips, (ii) the number of (left or right) rotations, and (iii) the key that appears in the root node immediately after the insertion.

Please note that the insertions should be cumulative

Q3.1

4.5 Points

Insert 18

(i) Number of color flips during insertion?

1

(ii) Number of rotations during insertion?

2

(iii) Key in the root after insertion?

13

Q3.2
3 Points

Insert 33

(i) Number of color flips during insertion?

0

(ii) Number of rotations during insertion?

1

(iii) Key in the root after insertion?

13

Q3.3
4.5 Points

Insert 45

(i) Number of color flips during insertion?

2

(ii) Number of rotations during insertion?

1

(iii) Key in the root after insertion?

19

Q4 Short Questions, Short Answers

8 Points

For each of the following pairs, briefly describe one reason why you'd use one instead of the other. ***Please do not just comment of their complexities.***

A familiar example is given below for Mergesort and Quicksort.

> One can prefer using Mergesort over Quicksort if he/she cares for stability. But, one can prefer using Quicksort over Mergesort if it is not possible to use extra space.

Q4.1

4 Points

Red-black tree vs. hashing

A red black tree is ordered, while a hashing is not. Red-black trees are more efficient for operations that depend on the ordering of elements.

Q4.2

4 Points

Sequential search vs. binary search

The main difference between linear search and binary search is that a binary search is more efficient and takes minimum time to search an element than a sequential search.

Q5

12 Points

0	1	2	3	4	5	6	7	8	9	10	11
	k_5			k_1	k_6	k_3		k_2	k_7		k_4

Given a linear probing hash table containing 7 keys as shown above, answer the following questions assuming that the hashes are uniformly distributed.

Q5.1

4 Points

Write down possible hash values for the key k_3.

4 or 5 or 6

Q5.2

4 Points

How many unique hash functions (different mappings of the keys above) can produce the array above.

12

Q5.3

4 Points

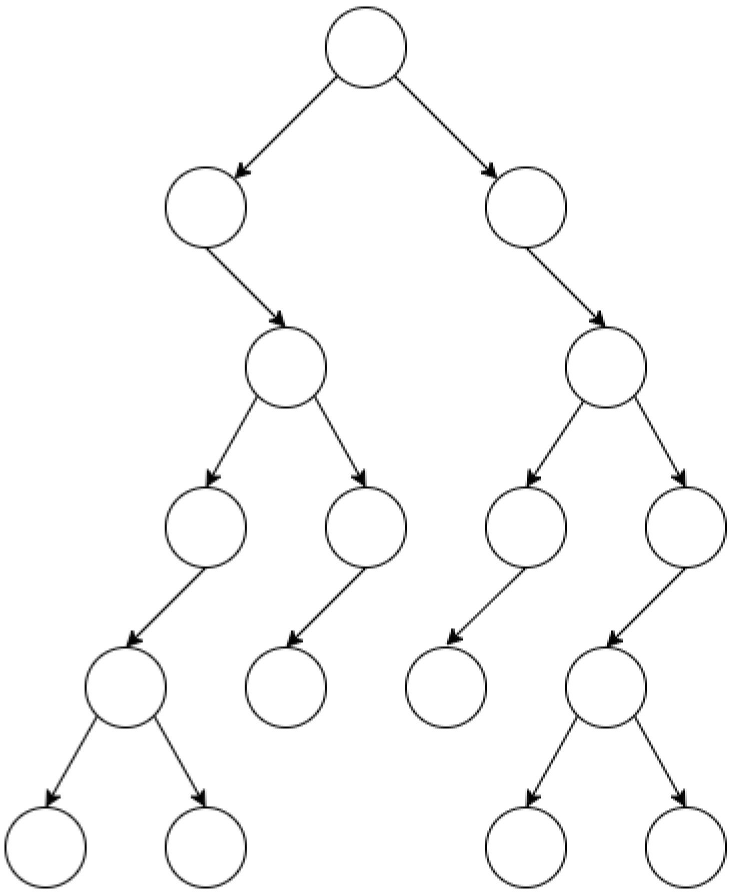
What is the probability of inserting a single new key that fills the 7^{th} index.

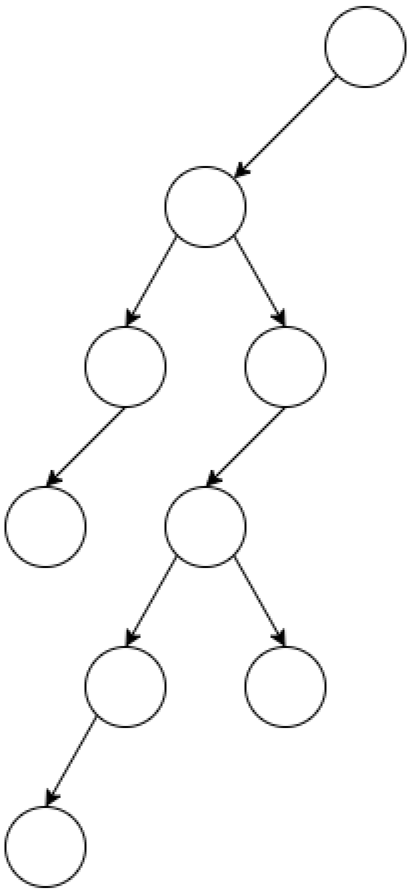
4/12 = 1/3 = 0.33

Q6

16 Points

Write a code that will check if a given binary tree has alternating regular levels, meaning that any two internal nodes in the same level of the tree has the same degree. Also, it is possible to have a leaf node in any level. For example, the two binary trees are alternating with regular levels, the first tree starts with degree 2, the second tree starts with degree one, and they both alternate between 1 and 2.





You can use the following class and define any number of functions.

```
private class Node
{
  private Key key;
  private Value val;
  private Node left, right;

  public Node(Key key, Value val) {
    this.key = key;
    this.val = val;
  }
}
```

```
Node FirstTree = new Node(Key,Value);
Node SecondTree = new Node(Key,Value);

public void printSameDegree(Node FirstTree,Node SecondTree){
if(FirstTree == null || SecondTree == null ){
return;
}
int first = 0;
int second = 0;
if(FirstTree.left != null){
first++;
}
if(FirstTree.right!= null){
first++;
}
if(SecondTree.right!= null){
second ++;
}
if(SecondTree.right!= null){
second ++;
}
if(first == second){
System.out.println("same degree "+FirstTree.key + " and " + SecondTree.key);
}

printSameDegree(Node FirstTree.right,Node SecondTree.right);
printSameDegree(Node FirstTree.right,Node SecondTree.left);
printSameDegree(Node FirstTree.left,Node SecondTree.right);
```

printSameDegree(Node FirstTree.left,Node SecondTree.left);
}

Q7 Algorithmic Complexity

23 Points

Suppose A is a given array of length n-1. Each element of A is an integer between 1 and n. All the elements of A are different. Consider the following function:

```
public static int something (int A[]) {
  int i, j, flag;
   for (j = 1; j <= A.length; j++) {
      flag = 0;
      for (i = 0; i < A.length-1; i++) {
        if (A[i]== j) {
           flag = 1;
           break;
        }
      }
      if (flag == 0)
         return j;
   }
   return -1;
}
```

Q7.1

5 Points

What is the worst-case time complexity of the function?

○ $O(n)$

○ $\Omega(n)$

○ $\Theta(n\, logn)$

[✓ $O(n^2)$]

Q7.2

18 Points

Write a function in JAVA that performs the same task in the worst-case $O(n)$ time complexity without introducing additional space

complexity.
Note that any solution with additional space or having time complexity less than the original one but greater than $O(n)$ will get partial credits.

```
public static int something(int A[], int counter) {
        int flag;
        for (int count = 0; count < A.length - 1; count++) {
            if (A[count] == counter) {
                flag = 1;
                break;
            } else {
                something(A, counter + 1);
                return counter;
            }
        }
        return -1;
    }

        something(A, 1);
```

Q8 Sorting

12 Points

Consider the follow array:
[33, -8, 42, 50, -4, 53, 11, 91, -79, 2]

Each of the following is a view of a sort in the progress of the above array. Choose the corresponding sorting algorithm for the given arrays.

- If the sorting algorithm contains multiple loops, the array is shown after a few passes of the outermost loop has completed.
- If the sorting algorithm is mergesort, the array is shown after the recursive calls have completed on each sub-part of the array.
- If the sorting algorithm is quicksort, the algorithm chooses the first element as its pivot. For quick sort, the array is shown before the recursive calls are made.

Q8.1

3 Points

[-79, -8, -4, 50, 42, 53, 11, 91, 33, 2]

[✓ Selection sort]

Insertion sort

Mergesort

Quicksort

Q8.2
3 Points

[2, -8, -79, 11, -4, 33, 50, 91, 42, 53]

Selection sort

Insertion sort

Mergesort

[✓ Quicksort]

Q8.3
3 Points

[-8, -4, 33, 42, 50, -79, 2, 11, 53, 91]

Selection sort

Insertion sort

[✓ Mergesort]

Quicksort

Q8.4
3 Points

[-8, 33, 42, 50, -4, 53, 11, 91, -79, 2]

- ○ Selection sort
- [✓ Insertion sort]
- ○ Mergesort
- ○ Quicksort

Midterm exam

● GRADED

STUDENT
Mehmet Taha Usta

TOTAL POINTS
50 / 100 pts

QUESTION 1

Academic Honesty	**1** / 1 pt

QUESTION 2

Heaps		**8** / 16 pts
2.1	(no title)	**0** / 4 pts
2.2	(no title)	**0** / 4 pts
2.3	(no title)	**4** / 4 pts
2.4	(no title)	**4** / 4 pts

QUESTION 3

Red-black trees		**12** / 12 pts
3.1	(no title)	**4.5** / 4.5 pts
3.2	(no title)	**3** / 3 pts
3.3	(no title)	**4.5** / 4.5 pts

QUESTION 4

Short Questions, Short Answers		**0** / 8 pts
4.1	(no title)	**0** / 4 pts
4.2	(no title)	**0** / 4 pts

QUESTION 5

	(no title)	**12** / 12 pts
5.1	(no title)	**4** / 4 pts
5.2	(no title)	**4** / 4 pts
5.3	(no title)	**4** / 4 pts

QUESTION 6

	(no title)	**0** / 16 pts

QUESTION 7

	Algorithmic Complexity	**5** / 23 pts
7.1	(no title)	**5** / 5 pts
7.2	(no title)	**0** / 18 pts

QUESTION 8

	Sorting	**12** / 12 pts
8.1	(no title)	**3** / 3 pts
8.2	(no title)	**3** / 3 pts
8.3	(no title)	**3** / 3 pts
8.4	(no title)	**3** / 3 pts