Student's Name	Number	Course Code	Instructor	Date	Signature
		CSE2046	Assist. Prof. Ö. Korçak	02.06.2020	

Q1	Q2	Q3	Q4	Q5	Q6	SUM
/13	/15	/24	/18	/25	/25	/100

CSE 2046 Analysis of Algorithms

Spring 2020

Online Homework 1

Solution and Submission Notes

- **Textbooks, slides, and notes are open.**
- **Internet usage is not allowed; however, computers and smartphones may be used for viewing course materials provided for this course.**
- **Write your name, surname and student ID on the top of each exam solution page.**
- **Show all your work.**
- **Write the following sentence on the top of the first page with your handwriting and sign it: "On my honor, I have neither given nor received any unauthorized assistance on this examination."**
- **Either print this PDF file (6 pages) or use blank A4 pages for your solutions. Solve all the questions with your handwriting.**
- **Scan all the solution pages to a single PDF file named "firstname_lastname.pdf". Do not include raw photos, use an Android/IOS scanner application. All the writings should be very readable. Page layouts should be portrait, not landscape.**
- **Upload the PDF file to https://ues.marmara.edu.tr system before the deadline. Give at least 15-20 minutes for scanning and upload process.**

Student's Name	Number	Course Code	Instructor	Date	Signature
		CSE2046	Assist. Prof. Ö. Korçak	02.06.2020	

On my honor,

Name Surname:_____

Signature:_____

Q-1. (a – 13 pts) For each of the following, indicate the class $\Theta(g(n))$ the function belongs to. Use the simplest $g(n)$ possible.

(i – 4 pts) $\sum_{i=1}^{2n} \log_2 i$

(ii – 3 pts) $\ln(\pi^{en}) + \ln(n^{\pi e})$

(iii – 3 pts) $\sqrt{(n-1)!} + 2^{\sqrt{n}}$

(iv – 3 pts) $10^{n/2} + n^2 3^n$

Q-2 (15 pts) Consider the following algorithm:

```
Algorithm HeapX:
Inputs: An unsorted array A[0..n − 1] of n keys; an integer k, s.t. 1 ≤ k ≤ n
Stage 1: Construct a heap for a given list of n keys

Stage 2: Repeat operation of root removal n − k times:
      Exchange keys in the root and in the last (rightmost) leaf
      Decrease heap size by 1
      If necessary, swap new root with larger child until the heap condition holds

Stage 3: Return the root key.
```

(a – 6 pts) What would be the output of HeapX?

(b – 9 pts) For each of the following, indicate whether it is "true" or "false". Give your reasoning. Answers without any comments will not be graded.

(i – 3 pts) Time complexity of HeapX is in $\Theta(n \log n)$.

(ii – 3 pts) Worst case time complexity of HeapX is asymptotically same as the average case time complexity of Heapsort.

(iii – 3 pts) Average case time complexity of HeapX is in $O(n \log n)$ and in $\Omega(n)$.

Student's Name	Number	Course Code	Instructor	Date	Signature
		CSE2046	Assist. Prof. Ö. Korçak	02.06.2020	

Q-3. (24 pts) Consider the following function.

```
void func1(int A[0..n-1]) {
int a;
if (n <= 1)
      printline("Take it easy!");
else {
      a = 1;
      while (a ≤ n/2){
            a = a * 2;
            func1 (A[a..a + ⌊n/2⌋ − 1);
      }
}
}
```

(a – 12 pts) Find a recurrence relation for number of lines the above function prints.

(b – 12) Solve the recurrence relation using backward substitution. (Find the exact value. You may assume that *n* is a power of 2)

Student's Name	Number	Course Code	Instructor	Date	Signature
		CSE2046	Assist. Prof. Ö. Korçak	02.06.2020	

Q-4. (18 pts) Consider the following function:

```
int func2(int A[0..n-1]) {
int a = 0;
if (n <= 1)
      return 1;
else {
      for (int i=0, i<=n-2; i++)
            for (int j=i+1, j<=n-1; j++)
                  for (int k=0, k<=⌊n/2⌋ − 1; k++)
                        a = a + 1;
      func2(A[0..⌊2n/3⌋-1);
      func2(A[⌊n/3⌋..n-1);
}
```

(a – 12 pts) Find the recurrence relation for the time complexity of func2. (Give the exact function.)

(b – 6 pts) Solve the recurrence relation using Master Theorem.

Q-5. (25 pts) Given the following:

- **An unsorted array A of *n* different character strings. Length of any string is at most *m*.**
- **A hash function *MD5_Hash(string)* that generates a hash code for any string. Generated hash code is a bit string of 16 bytes (the output of the *MD5_Hash* function has always the same length). Time complexity of *MD5_Hash* function is $\Theta(m)$ where m is the length of the input string.**
- **An unsorted array C of *k* different hash codes. Length of each hash code is 16 bytes as told above.**

The problem is to determine number of strings in A whose hash codes exist in array C.

Here is an example:

Let's say A = {"hello", "take", "it", "easy"}, and:
MD5_Hash("hello") = 5D41402ABC4B2A76B9719D911017C592 (This is a hexadecimal representation of a bit string of 16 bytes. Each hexadecimal number represents 4 bits.)
MD5_Hash("take") = 93E5D1D0E8C4DA0F12F5BAE2C2FCED78
MD5_Hash("it") = 0D149B90E7394297301C90191AE775F0
MD5_Hash("easy") = 48BB6E862E54F2A795FFC4E541CAED4D
And if C = {C39436EE452E641CDE2EB992AB397911, 93E5D1D0E8C4DA0F12F5BAE2C2FCED78, A16AB4E64CE10E020D2A26947AA75B4A, 5D41402ABC4B2A76B9719D911017C592}

Then the answer would be 2 because hash codes of "hello" and "take" are in C.

(a – 10 pts) Design a brute-force algorithm for this problem. Describe its time complexity (as a function of *m*, *k* and *n*).

(b – 15 pts) Design a more efficient algorithm based on binary search. Describe its time complexity (as a function of *m*, *k* and *n*).
Note: Give a step-by-step description. Don't give a pseudocode.

Q-6. (25 pts) Consider that you have a bit string A consisting of *n* bits. Your goal is to count number of occurrences of 010 in this bit string.

Examples:

input: A=00000000000000000001000000; Output:1

input: A=01010101010; Output:5

(a – 10 pts) What is the number of comparisons in the worst case, if you apply brute-force string matching algorithm? (Give exact count in terms of *n*. Also give the form of A in the worst case.)

(b – 15 pts) Design a divide and conquer algorithm. Give a step-by-step description. What is the worst case time complexity in Θ form? (Give also the recurrence relation).

Notes: No grades will be given to solutions other than divide & conquer. Give step-by-step description instead of a pseudocode.