Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Assignment 4 (3% - 12 points)

CSI2110/CSI2510 (Fall 2021)

Due: Thursday Oct 14, 11:59PM

Late assignment policy : ***1min-24hs late are accepted with 30%off; no assignments accepted after 24hs late.***

Question 1. **[6 points]**

Given the following minheap, stored in an array *A*, transform it into a maxheap using the bottom-up heap construction algorithm covered in the class that runs in linear time.
The algorithm is as follows and fill (update) the table below for each *i*.

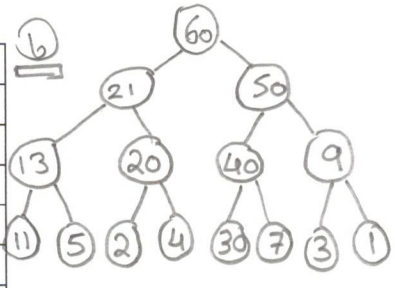
```
For (i=(n-2)/2; i>=0;i--)
{
    downheap(A,i);  /* downheap for a maxheap */
   /*  update the table A below */
}
```

Array A:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Keys of A	**1**	**2**	**3**	**5**	**4**	**7**	**9**	**11**	**13**	**20**	**21**	**30**	**40**	**50**	**60**

(3 points) a) Show the **array after each iteration** of the loop.

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Keys of A (before the loop)	**1**	**2**	**3**	**5**	**4**	**7**	**9**	**11**	**13**	**20**	**21**	**30**	**40**	**50**	**60**
Keys of A	1	2	3	5	4	7	60	11	13	20	21	30	40	50	9
Keys of A	1	2	3	5	4	40	60	11	13	20	21	30	7	50	9
Keys of A	1	2	3	5	21	40	60	11	13	20	4	30	7	50	9
Keys of A	1	2	3	13	21	40	60	11	5	20	4	30	7	50	9
Keys of A	1	2	60	13	21	40	50	11	5	20	4	30	7	3	9
Keys of A	1	21	60	13	20	40	50	11	5	2	4	30	7	1	9
Keys of A	60	21	50	13	20	40	9	11	5	2	4	30	7	3	1



(1 point) b) Draw the **final tree**.

PRACTICE EXERCISE: Do heapsort on this example (first part already done in 1a). No marks.

(1 point) c) Go back to the original **min-heap** in Array A:
Array A:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Keys of A	**1**	**2**	**3**	**5**	**4**	**7**	**9**	**11**	**13**	**20**	**21**	**30**	**40**	**50**	**60**

Show the **array after insert(3)**.

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Keys of A	1	2	3	3	4	7	9	5	13	20	21	30	40	50	60	11

(1 point) d) Go back to the original **min-heap** in Array A:
Array A:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Keys of A	**1**	**2**	**3**	**5**	**4**	**7**	**9**	**11**	**13**	**20**	**21**	**30**	**40**	**50**	**60**

Show the **array after removeMin()**.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Keys of A	2	4	3	3	20	7	9	5	13	60	21	30	40	50	60

Question 2. **[6 points]**

With a given array of n positive integers, make an algorithm using peudo-code to find kth smallest key element where $k \leq n$.
Example: For A=[77,15,2,300,75,50,20,97] and k=3, the answer is 20 because the 1th smallest is 2, 2nd smallest is 15, 3rd smallest is 20.

(i) [3 points] Describe a solution using min-heap with total complexity $O(n + k \log(n))$. Decribe the idea or give a pseudo-code. Java code is not allowed. Briefly explain why you achieve the required big-Oh.

(ii) [3 points] Describe a solution using a max-heap containing k elements. The complexity must be $O(n \log(k))$. Decribe the idea or give a peudo-code. Java code is not allowed. Briefly explain why you achieve the required big-Oh.

Question 2:

[i] I will describe the idea.

By using a min-heap we know that the root will be our minimum value. and if we want the first smallest element we would just removeMin() once and we would get the smallest elemat. since the root now is updated it will be the second smallest elemat.

So the idea is to removeMin() K times to get the K smallest element

Putting it in Min-Heap would be O(n) and each time we remove min it would be K log(n)

So the total complexity would be O(n + K log(n))

[ii] By using a max-heap of siz K to get the smallest K, our root in that case would be the K smallest,

as we only update the max heap if the elemant inserted is smaller than the nat and by laws of max heap the parent need to be larger than children. so getroot would give us the K-smallest value

and since this time we know that the value we are looking for is the root so getting it would be O(1) and building the max heap of size K would be O(n log(K))