ASSUMPTION UNIVERSITY

Vincent Mary School of Science and Technology
Department of Computer Science
Department of Information Technology

Midterm Examination
Semester 2/2021

Subject Code :	CS3003/IT2230/ITX2010, BIS4787
Subject Title :	Data Structure and Algorithms, Data Structure
Date :	January 21, 2022
Time :	**12.00 – 14.00 (2 hours)**
Instructors :	Asst. Prof. Dr. Thitipong Tanprasert (Full-Time)

Instructions:

1. Read the questions carefully and answer each question completely, legibly, and concisely.
2. You must type/write your answers in your computer only. Writing answers in a paper and taking a photo using camera are not allowed.
3. An answer to a question is either a text file, or docx file, or a Python 3 program.
4. To submit your answers, compress (zip) all of your answers into one file and upload the zipped file to the Midterm Examination created as an assignment for the course no later than 14:00 the latest. Any late submission received after 14:01 will not be graded.
5. This is an opened book examination; you can use any materials as references, including online search. However, any form of communication with anyone regarding the exam, directly or indirectly, will be considered "CHEATING".
6. You MUST turn on your camera and microphone, share your working screen (the entire screen of your PC), and record the video in MS Team for the whole examination period. The answered file will NOT BE GRADED IF THERE IS NO COMPLETELY RECORDED VIDEO CLIP.
7. This examination paper and recorded video are an intellectual property of Assumption University; you are NOT allowed to duplicate, share, or publicize it.
8. If you cheat or contribute to cheating at the exam, you will get zero score and will be considered to get the grade 'F' for this course.

Marking Scale:

Essay and/or Programming	5 questions	60 marks

Total 60 marks

Total 3 pages (including this page)

1) [10 marks] Determine an upperbound on the running time of the following codes.

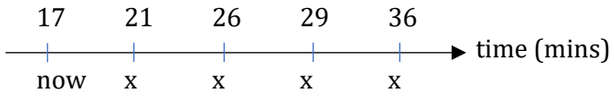
`# A is an arbitrary list of numbers`
 	
`for i in range(n-1, -1, -1):`
`  # get_max(i) return index of the max value in A[0..i]`
`  j = get_max(i)`
`  A[i],A[j] = A[j],A[i]`
	T(n) = O(_____)

`h = heap(A, morethan)  # build max heap`
 	
`for i in range(n-1, -1, -1):`
`  A[i] = h.extract_max()`
	T(n) = O(_____)

If the running time of the two codes above are different, which one is faster? And what is the reason that it is faster?

2) [20 marks] Visa application reservation.
- Reservation is required for the time to submit Visa application document and interview.
- When the application is submitted, the reservation is removed from set of pending events
- "Reserve request" specifies the requested time, t, to get service from the Visa office.
- t will be added to the list of reservations if no other previous reservations are scheduled within k minutes both before and after t.

Example



If k = 3,
- 24 is not OK (must be at least 3 minutes away from 26)
- 33 is OK
- 10 is invalid (already past)

Given that the accepted reservation list is R = [R[0], R[1], R[2], … , R[n-1]], where R[i] is the requested time of reservation i.

You are to write a Python program that takes R and a new request time, t, as input. Then, the program either add this new reservation to the list or reject, depending on whether the criteria for reservation is met.

a) [5 marks] Apply the insertion sort technique to keep the list R sorted all the time. Insert a new reservation into R only if it is at least k minutes away from both the reservations before and after t.

b) [5 marks] If the length of R is n, what is the upperbound on the running time of the algorithm for proessing a new request in question a) above?

c) [5 marks] As an alternative, if you store the reservation list R in a binary search tree, suggest the process of how t is to be determined whether it can be added into the list.

d) [5 marks] Following the question c) above in which a binary search tree is utilized, what is the upperbound on the running time for processing a new request?

3) [10 marks] A quicksort code, in Python, is given as

```
def quicksort(p, r):
    global A

    if p < r:
        q = partition(p,r)
        print(A[p:q], A[q], A[q+1:r+1])
        quicksort(p,q-1)
        quicksort(q+1,r)
```

Let A = [52, 37, 63, 14, 17, 8, 6, 25]. What are the output of quicksort(0, n-1)?
You can assume any valid partitioning algorithm.

4) [10 marks] A mergesort code, in Python, is given as

```
def mergesort(A, p, r):
    if p < r:
        q = (p+r)//2
        mergesort(A, p,q)
        mergesort(A, q+1,r)
        merge(A, p, q, r)
        print(A[p:r+1])
```

Let A = [52, 37, 63, 14, 17, 8, 6, 25]. What are the output of mergesort(A, 0, n-1)?

5) [10 marks] Given a list of n distinct integers, write a Python program to find if there are two pairs (a, b) and (c, d) such that a*b = c*d, where a,b,c,d are all distinct integers. If there are multiple pairs that meet the criterion, print any of them.

INPUT: A sequence of distinct integers, separated by space.

OUTPUT: If the pairs exist, print the first two numbers (of the first pair), followed by a comma, followed by the second two numbers (of the second pair).
If no pair exists, print "No pair exists".

Example

INPUT	OUTPUT
7 4 3 1 8 9 6	3 8 , 4 6
30 65 1 90 8 9 7	No pair exists

The program must be able to handle a list of 2000 integers within 1 second.

Hint: Use the provided code template.
Requiring a technique for fast searching is inevitable.