CS2040S 2020/2021 Sem 1 Midterm

MCQ: 30 Marks, 10 Questions (3 marks each)

1. What is the time complexity of the following pseudo-code:

```
1. initialize boolean array A of size N+1 (N>2) to true

2. for (int i=2; i <= N; i++)
     for (int j=2; j*i <= N; j++)
       A[j] = false;
```

a. O(N)
b. O(1)
c. O(N^2)
d. O(NlogN)

2. Given two **nonempty** data structures X and Y, both containing elements in ascending order (the first element to be removed from the DS is the smallest element in the DS), we want to merge the contents of Y into X, such that X contains the elements of both X and Y, in ascending order. **Elements are distinct across both X and Y (so a value will appear at most once in X and Y combined). You should do this in O(m+n) time, where m and n are the number of elements in X and Y respectively.** You are not allowed to use any other DSes, apart from the provided X and Y, and are not allowed to use recursion. This task is possible if:

i. X is a queue and Y is a queue

ii. X is a stack and Y is a stack

iii. X is a queue and Y is a stack

iv. X is a stack and Y is a queue

a. i and ii
b. i, iii and iv
c. ii only
d. ii and iv

3. Given an unsorted array of N distinct integers with values from the range [1..N], we can find the kth smallest element in:

a. O(1)
b. O(logN)
c. O(N)
d. O(NlogN)

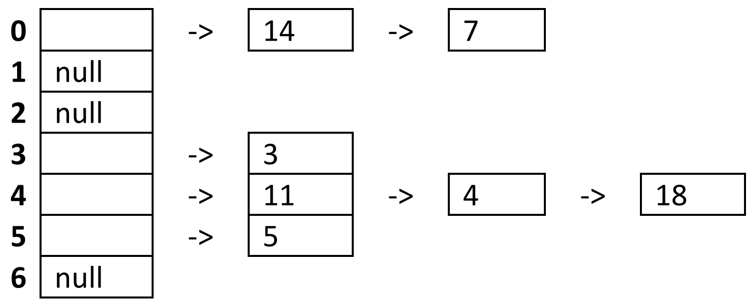
4. We want to implement a new ADT, which supports the 4 operations from the Queue ADT (**offer()**, **poll()**, **peek()**, **isEmpty()**), as well as a new operation, **cut(x)**, which removes the first x elements of the queue and keeps the other elements. The **cut(x)** method should run in O(1) time, while the original Queue ADT methods should still have a worst-case time complexity of O(1). You may assume that if you use an array, the number of valid elements in the ADT will always be less than the array size, at any point in time. The most appropriate data structure to do so is:

a. Circular Linked List (tailed linked list with tail pointing to head)
b. Doubly Linked List with tail reference
c. Array
d. Circular Array

5. Given a sorted array (containing non-distinct integers), we want to find out how many times a certain integer appears in the array. We can do so in:

a. O(logN)
b. O((logN)^2)
c. O(N$^{0.5}$)
d. O(N)

6. Given the following hash table using separate chaining:



The load factor of this table (rounded to 2 decimal places) is:

a. 0.57
b. 1.00
c. 1.43
d. 1.75

7. A bloom filter is created with the following hash functions:

h_1 = (key % 37) % 11

h_2 = (key % 31) % 11

h_3 = (key % 23) % 11

Below is the bloom filter after inserting keys 142 and 97.

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

We then attempt to check if the following keys are present in the bloom filter. Which of these keys would give a negative result (i.e return false)?

i. 287

ii. 142

iii. 384

iv. 545

a. i and iii
b. ii only
c. i only
d. i,iii and iv

8. Given the following method in pseudo-code:

```
 // q  is a Queue that contains integers
void foo(Queue q) {
  i = 0;
  while(!q.isEmpty()) {
    if (q.poll() == q.poll()) {
      q.offer(i++);
    }
  }
}
```

Assuming the queue contains [5, 6, 6, 7] initially, which of the following options is correct?

a. q is empty after the while loop ends
b. q = [0] after while loop ends
c. method will cause a runtime error if implemented
d. method goes into Infinite loop

9. Given an array split into half (you may assume the array size is always a multiple of 2), where the first half contains elements in sorted order, and the second half contains elements in any order (may be unsorted), choose the most efficient algorithm in terms of worst case time complexity to sort this array.

a. Insertion sort
b. Selection sort
c. Merge sort
d. Quick sort

10. The following hash table was constructed with the hash function h(key) = key % 11. No deletions were involved in the construction of this hash table:

0	1	2	3	4	5	6	7	8	9	10
null	null	4	null	15	48	null	29	62	null	21

The collision resolution technique could be:

i. Modified linear probing with step size d = 3
ii. Quadratic probing
iii. Double hashing with second hash function g(key) = 7 - (key % 7)

a. i only
b. ii only
c. ii and iii
d. i, ii and iii

Analysis: 18 marks, 3 questions

Choose the correct answer (2 marks) and give your reasons for it (4 marks)

11. A bloom filter cannot be designed such that the false positive rate is 0%, even if we have infinite memory.

    a. True
    b. False

12. Other than an input of keys all in descending order, there is no other input that will cause optimized bubble sort to run in O(N^2) worst case time.

    a. True
    b. False

13. Given the following code fragment, the worst case time complexity of calling f1(N) for some N that is a power of 3 is O(N).

```
void f1(int n) {
  if (n >= 1) {
    for (int i=0; i < 10; i++) {
      f2(n);
    }
    f1(n/3);
    f1(n/3);
    f1(n/3);
  }
}


void f2(int m) {
  for (int j = 0; j < m; j++) {
    System.out.println("Hello!");
  }
}
```

a. True
b. False

Structured Questions: 52 marks, 7 questions

Write in **pseudo-code**.

Any algorithm/data structure/data structure operation not taught in CS2040S must be described, there must be no black boxes.

Partial marks will be awarded for correct answers not meeting the time complexity required.

Questions 14 to 16 refer to the following problem description

New operations for Linked List and Stack

We want to include new operations to the Link List and Stack ADT. Assume that the Link List and Stack contains integer values.

The first new operation is a Stack operation and is as described:

Rotate(Stack s, int dir, int m) -

The Stack s will be rotated in the direction given by dir a total of m times. dir == 1 means rotate left, dir == 2 means rotate right.

Example of stack rotation:

Given s = 1,2,3,4,5   where 1 is the top of the stack and 5 the bottom of the stack

Rotate(s,1,1) will rotate s left 1 time and change s to be s = 2,3,4,5,1

Rotate(s,1,3) will rotate s left 3 time and change s to be s = 4,5,1,2,3

Rotate(s,2,1) will rotate s right 1 time and change s to be s = 5,1,2,3,4

Rotate(s,2,3) will rotate s right 3 time and change s to be s = 3,4,5,1,2

14. Give the algorithm for **Rotate** using only the Stack operations empty(), push(), pop(), peek() and size() in **O(n)** worst case time, where n is the size of the stack. You can use at most 2 additional stack to help you and no other DSes (no arrays, LLs, queues etc ...).

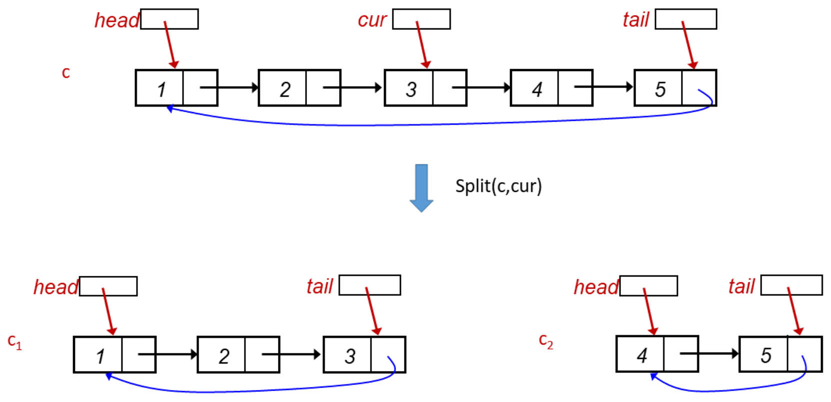
[8 marks]

The second new operation is a circular linked list (**A tailed singly linked list where tail.next points to head**) operation and is as described:

TailedLinkedList[] Split (TailedLinkedList c, ListNode cur) -

Given cur which points to a node of in a circular linked list c, split c into 2 circular Linked Lists.
The first one c_1: Containing head of c to cur and the head of c_1 being the head of c.
The second one c_2: Containing nodes from cur.next to tail of c,with cur.next being head of c_2.
If cur points to tail of c then return null, otherwise store c_1 and c_2 in an array of size 2 and return that array.

Example of a split where cur is not pointing to tail of c:



15. Give an algorithm to perform **Split** by only manipulating references in the original circular linked list c and having head and tail of c_1 and c_2 pointing to the correct nodes. You should not create any new nodes. You algorithm should run in **O(1)** worst case time. **[6 marks]**

The third new operation is a basic/singly linked list operation and is as described:

DeleteSecondLast (BasicLinkedList c, ListNode cur) -

Given cur which points to the second last node of c, delete that node.

16. Give an algorithm to perform **DeleteSecondLast** in **O(1)** worst case time. You may assume that c always has ≥ 2 nodes before the second last node is deleted.

[4 marks]

17. Given an array A of N unsorted and unique integers between 0 and 100,000,000, find the total number of pairs x+y such that x+y = z.

    You cannot use any form of hashing in your algorithm and it must run in O(N) worst case time.

[7 marks]

18. Given 2 unsorted arrays, **A** containing N integers with values ranging from 0 to N^N and **B** containing M integers with values ranging from 0 to M^M, give the best algorithm (in terms of average time complexity) you can think of that will output true if B is a subset of A and false if it is not. Both A and B might have duplicate values.

[7 marks]

Questions 19 to 20 refer to the following problem description

Lots of points

Array A contains N points where each point consist of 2 integers x and y representing the x and y coordinate of the point (1 ≤ x,y ≤ 1,000,000,000).

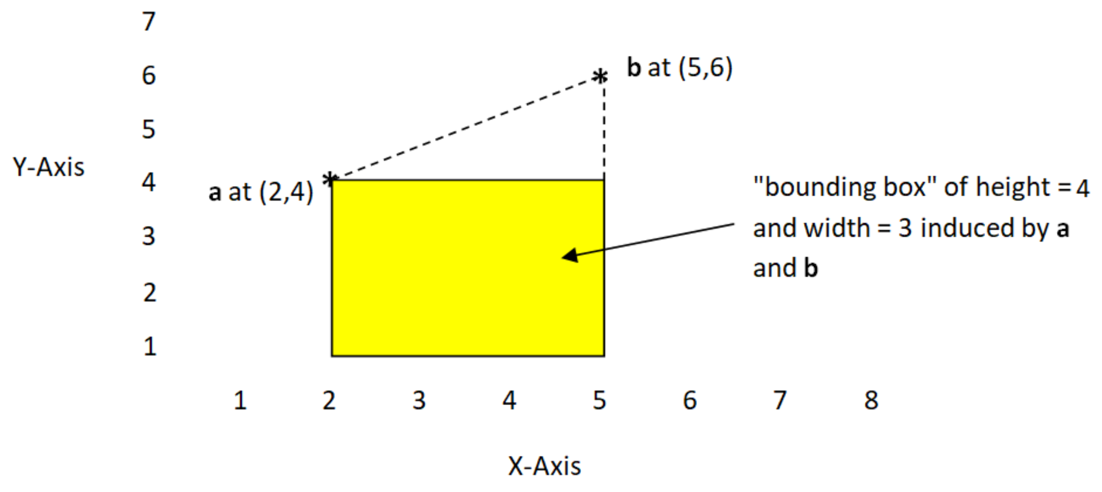
These N points in A is neither sorted by x-coordinate or y-coordinate.

Given any 2 points **a** and **b** there is a "bounding box" induced by **a** and **b**.

Height of bounding box = min(**a**.y,**b**.y)

Width of bounding box = |**a**.x-**b**.x|

This is illustrated diagrammatically below

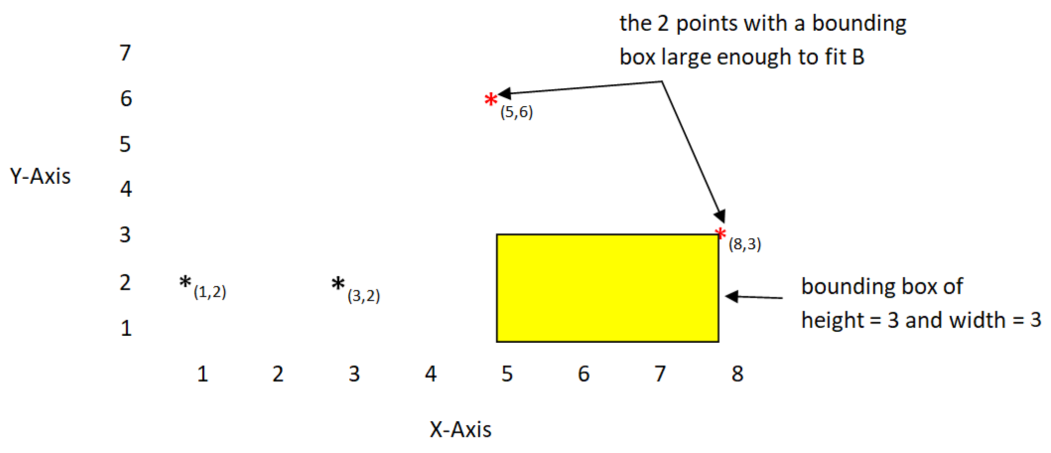

Y-Axis
X-Axis
a at (2,4)
b at (5,6)
"bounding box" of height = 4 and width = 3 induced by **a** and **b**
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8

19. Now given a rectangular block B of height **h** and width **w**, return 1 pair of points in A such that B can fit within the bounding box induced by the pair of points (without rotating B). If there is no such pair, output "impossible".

Your algorithm must run in O(N) worst case time.

[8 marks]

An example is shown below



the 2 points with a bounding box large enough to fit B
7
6
5
4
3
2
1
Y-Axis
(5,6)
(8,3)
(1,2)
(3,2)
bounding box of height = 3 and width = 3
1
2
3
4
5
6
7
8
X-Axis

Given a rectangular block B of height = 3 and width = 3, B can fit into the bounding box induced by the 2 points as indicated, and those are the only 2 points with a bounding box large enough to fit B

20. Now with pre-processing that does not exceed $O(N)$ worst case time, design a $O(logN)$ worst case time algorithm that given the height **h** and width **w** of any block B, will return a pair of points in A with a bounding box large enough to fit B. If there is no such pair output "impossible".

    Describe your algorithm for preprocessing (including any DS used) and also the algorithm to answer each query.

[12 marks]