

Family Name \_\_\_\_\_

First Name \_\_\_\_\_

Student Number

Venue \_\_\_\_\_

Seat Number \_\_\_\_\_



**No electronic/communication devices are permitted.**

**No exam materials may be removed from the exam room.**

## Computer Science and Software Engineering EXAMINATION

End-of-year Examinations, 2020

### COSC122-20S2 (C) Introduction to Computer Science

**For Examiner Use Only**

Question                      Mark

**Examination Duration:**                      120 minutes

**Exam Conditions:**

Closed Book exam: Students may not bring in any written or printed materials.

Calculators with a 'UC' sticker approved.

**Materials Permitted in the Exam Venue:**

None.

**Materials to be Supplied to Students:**

1 x Write-on question paper/answer book

**Instructions to Students:**

Answer all questions. This exam is worth a total of 100 marks.

All programming questions are to be taken as referring to Python v3.8. Where Python code is shown, the numbers on the left side indicate line numbers. Check carefully the number of marks allocated to each question. This suggests the degree of detail required in each answer and therefore the amount of time to spend on it. The amount of space provided also indicates the amount of detail expected.

Mark or write in the spaces allocated to each answer. Do not write close to the margins, as the answer books will be scanned, and writing very close to the margin may not be picked up. If you require extra room there are blank pages at the end of this booklet. You may also use additional sheets of paper; these must be fastened securely to your answer booklet. You should clearly indicate in the appropriate space that the answer is continued/provided elsewhere.


**Questions Start on Page 3**

1.

[Question total: 8 marks]

Give the “big-oh” asymptotic complexity for the following:

- (a) [1 mark] The average case number of comparisons needed by sequential search on a list of  $n$  keys.

- (b) [1 mark] The best case time to find the minimum value in a list of  $n$  keys that are in random order.

- (c) [1 mark] An algorithm that uses  $n - 2 + 2n \log_4 n$  operations.

- (d) [1 mark] The best case time for performing a quicksort *partition* on a list of  $n$  items (not the full quicksort).

- (e) [1 mark] Pushing an item on a stack of  $n$  items, assuming the stack pushes items at location 0 of a Python list (i.e. `a_stack.insert(0,item)` )

- (f) [1 mark] The worst case time for the fast `build_heap` (`heapify`) method, which turns an unsorted list into a heap.

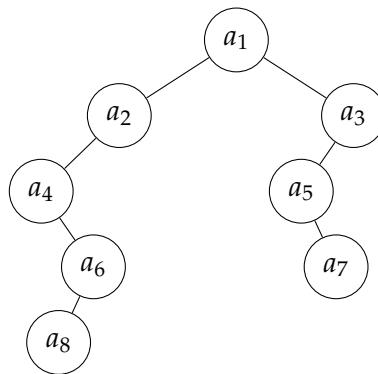
- (g) [1 mark] The worst case depth of a binary search tree.

- (h) [1 mark] The worst case time to find the maximum value in a min-heap.

2.

[Question total: 7 marks]

The valid Binary Search Tree below contains values denoted by  $a_1$  to  $a_8$ .



(a) [1 mark] Which node contains the smallest value in the tree?

(b) [1 mark] Which node contains the *second* smallest value in the tree?

(c) [2 marks] Which node is the in-order successor to node  $a_1$ ?

(d) [3 marks] Give the order that nodes would be visited in an in-order traversal of the tree.

3.

[Question total: 6 marks]

(a) [2 marks] How many comparisons will selection sort make in the best case to sort a list of 20 keys?

- (b) [2 marks] How many comparisons will insertion sort make in the best case to sort a list of 20 keys?

- (c) [2 marks] How many comparisons will the 2-sort part of a Shell sort use on a list that has a total 20 keys, in the best case? (This is about the 2-sort phase from Shellsort, not a full sort.)

4. [3 marks]

[Question total: 3 marks]

The function “mystery” uses a standard stack. What does the function do?

```
def mystery(my_list):  
    s = Stack()  
    for item in my_list:  
        s.push(item)  
    while not s.is_empty():  
        print(s.pop())
```

5.

**[Question total: 6 marks]**

Consider quicksort where the partition operation uses the *first* value in a list as the pivot.

- (a) **[2 marks]** What is the running time of quicksort if the list is already sorted in increasing order? Use big-oh notation. Explain briefly.

- (b) **[2 marks]** Answer the same question as in part (a) above, but now assume we call quicksort on a list that is already sorted in *decreasing* order. Explain briefly.

- (c) **[2 marks]** Now assume that the partition operation, instead of using the first element as the pivot, uses the median-of-three as pivot. What is the running time of quicksort with this partition when called on a list sorted in increasing order? Use big-oh notation. Explain briefly.

6.

**[Question total: 15 marks]**

The following Python method is for percolating up (sifting up) in a binary heap that is stored in a list that starts at location 1:

```
1 | def perc_up(self,i):  
2 |     while i // 2 > 0:  
3 |         if self.heap_list[i] < self.heap_list[i // 2]:  
4 |             temp = self.heap_list[i // 2]  
5 |             self.heap_list[i // 2] = self.heap_list[i]  
6 |             self.heap_list[i] = temp  
7 |         i = i // 2
```

(a) **[2 marks]** Is the code for a max-heap or min-heap? Explain how you can tell.

(b) **[2 marks]** Explain the meaning of line 2 in terms of what is happening if the heap is thought of as a tree.

(c) **[2 marks]** Explain the meaning of lines 4 to 6 in terms of what is happening if the heap is thought of as a tree.

(d) **[3 marks]** In the worst case, approximately how many times is line 4 executed if `perc_up` is passed the value *i*?

- (e) [3 marks] In the best case, how many times is line 4 executed if `perc_up` is passed the value  $i$ ?

- (f) [3 marks] If the heap was a ternary heap (i.e. 3-heap), several changes to this code would be needed. What would line 7 need to change to for a 3-heap, assuming that the root is at index 1?

7.

[Question total: 6 marks]

- (a) [3 marks] Show a parse tree for the infix expression:

$$(a \times (b + c)) - (d \times e)$$

- (b) [3 marks] Give the postfix notation of the above expression (you don't need to show your working).



8.

**[Question total: 11 marks]**

Merging can be done on more than two lists at a time. For example, to merge the following three lists in one pass, the three values that start each list respectively are compared to find the smallest, which is removed from its list and appended to the merged result; this is repeated until all the items in the three lists have been transferred to the merged list. If one of the three lists is emptied, the merging continues with the other two.

[2, 3, 6, 9]  
[1, 4, 8, 10]  
[5, 7, 11, 12]

- (a) **[2 marks]** How many key comparisons are required to find the smallest of the items at the front of the three lists?

- (b) **[3 marks]** How many key comparisons in total are needed to merge these three lists?

- (c) **[3 marks]** Give three lists, each containing four items, that would give the *best* case number of comparisons if they were merged this way.

- (d) **[3 marks]** Suppose instead of three lists there are  $k$  lists to merge. Finding the smallest of the  $k$  items at the front of the lists could get cumbersome, as it is repeated over and over. As a step to finding a faster way to find the smallest of the  $k$  items, what is the abstract data type that is used in this situation? (Explain)

9.

**[Question total: 6 marks]**

Suppose the quicksort partition operation is performed on the following list, with the first item (18) being used as the pivot:

[18, 34, 2, 4, 45, 12, 10, 6]

- (a) **[2 marks]** How many key comparisons will the partition make?

- (b) **[4 marks]** Next, quicksort would normally recursively partition both the left and right side of the partitioned list. However, if the partition was being used as part of an algorithm to find the five smallest items in the list in increasing order (i.e. to give the answer 2, 4, 6, 10, 12), explain which partition operations are still needed to produce this solution.

10.

**[Question total: 3 marks]**

- (a) **[1 mark]** Which data structure is better suited to binary searching, a list (i.e. an array type structure, such as the Python list data type) or a singly-linked list?

- (b) **[2 marks]** Give one advantage that chaining has over open-addressing; and one advantage that open-addressing has over chaining.

Advantage of chaining:

Advantage of open addressing:

11.

[Question total: 5 marks]

The following code is part of a Python declaration for a linked list, but it contains a bug!

```
1| class Node:
2|     def __init__(self, initdata):
3|         self.data = initdata
4|         self.next_node = None
5|
6| class LinkedList():
7|     def __init__(self):
8|         self.head = None
9|
10|    def add_item(self, item):
11|        temp = Node(item)
12|        self.head = temp
13|        temp.next_node = self.head
14|
15|    def print_list(self):
16|        current = self.head
17|        while current is not None:
18|            print(current.data)
19|            current = current.next_node
```

(a) [3 marks] When the following code is run the program goes into an infinite loop.

```
my_list = LinkedList()
my_list.add_item(7)
my_list.add_item(42)
my_list.add_item(122)
my_list.print_list()
```

Why does this happen? (You can use diagrams to help illustrate your answer if you want.)



(b) [2 marks] How would you fix the bug in this declaration?



12.

[Question total: 7 marks]

Consider the following recursive function:

```
1| def rec(val):  
2|     if val == 0:  
3|         print(val)  
4|     else:  
5|         print(val)  
6|         val = val//2  
7|         rec(val)
```

(a) [2 marks] What values will be printed out if the function is called with `rec(8)`?

(b) [5 marks] Give the three key characteristics/requirements of a recursive algorithm (i.e. the three laws of recursion), and the line number(s) where each of these appear in the above program.

1.

2.

3.

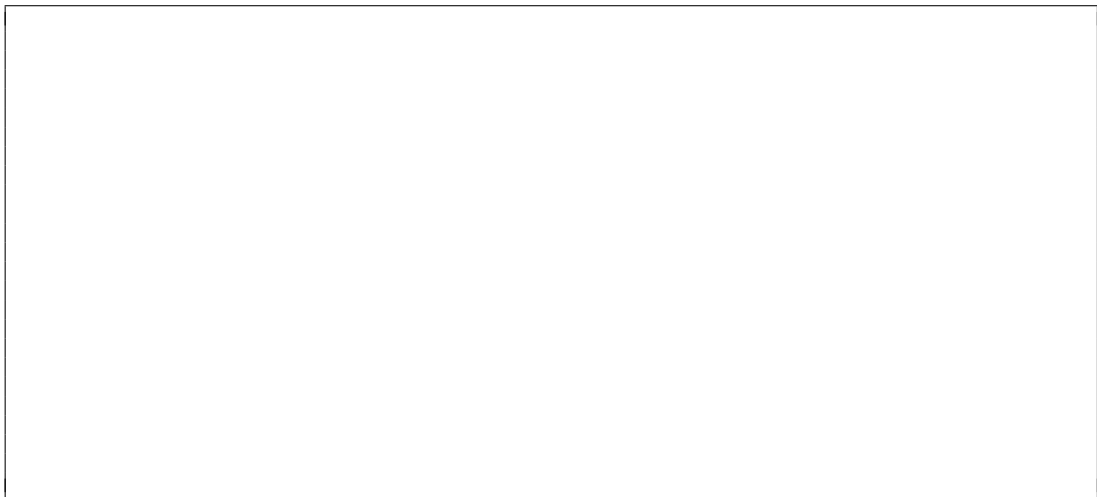
13.

[Question total: 6 marks]

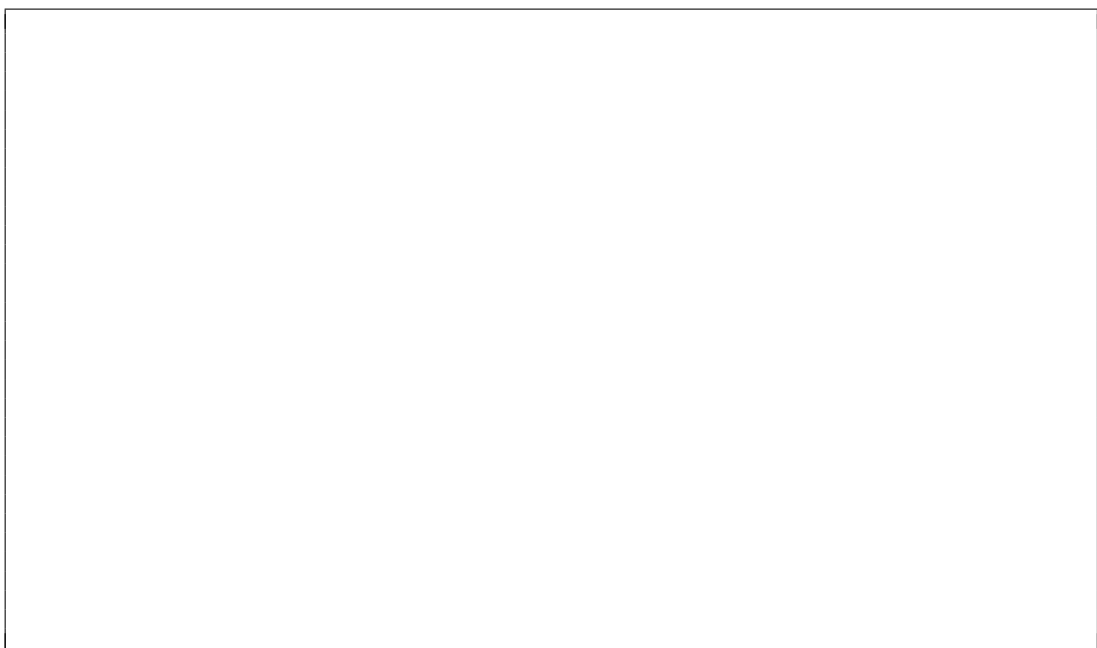
The following adjacency list represents a graph:

A | {C, 7}, {D, 3}  
B | {A, 9}  
C | {E, 7}, {F, 1}  
D | {B, 1}, {C, 8}  
E |  
F | {D, 2}, {G, 5}  
G | {B, 4}

- (a) [2 marks] Draw the diagrammatic representation for the graph in the adjacency list above.

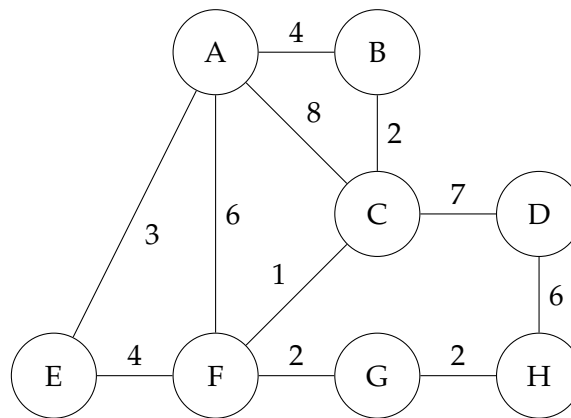


- (b) [4 marks] For the graph above, list the order that vertices are visited during a breadth first search (BFS) starting from vertex **A**, and show the state of the main data structure (stack or queue) before each node is visited. Make sure you clearly indicate the front/top and rear/bottom of the stack/queue. If you have a choice of vertices to add then they should be pushed/enqueued in alphabetical order.



14. Use the following undirected graph to answer these questions.

[Question total: 8 marks]



- (a) [4 marks] Give the shortest path tree for the above graph, with vertex A as the source. Label each node with the letter and cumulative weight, separated by a colon, eg, Z:26

- (b) [1 mark] If the weights in this graph represent distances, and edges are traversed along the shortest known path from vertex A, which vertex is furthest from vertex A?

- (c) [3 marks] Give a Minimal spanning tree (MST) for the graph.

15. [3 marks]

[Question total: 3 marks]

This question is based on the following adjacency matrix.

	a	b	c	d	e	f	g	h	i
a		1		1		1	1		
b	1		1	1	1	1	1		1
c		1			1		1		
d	1	1			1			1	1
e		1	1	1			1		
f	1	1							1
g	1	1	1		1				
h				1					
i		1		1		1			

Does this graph contain an Euler path? Explain how you know this.

— END OF EXAM —

...extra space ...

If you use this page please refer to it from the original question.



...extra space ...

If you use this page please refer to it from the original question.

...extra space ...

If you use this page please refer to it from the original question.