EXAMINATION PAPER: ACADEMIC SESSION 2020/2021

Campus	Maritime Greenwich
Faculty	Liberal Arts and Sciences
School	Computing and Mathematical Sciences
Level	4
MODULE TITLE	Algorithms and Data Structures
MODULE CODE	COMP1819
Date and Time	May 2021 - 90 minutes

Answer **ALL** questions

This is a multi-choice, open-book examination. You may access the internet but you may not communicate in any way with another person (including by electronic means).

To give your answers, you must:

- **Submit your answers on Moodle.**
- Also, make sure to mark your choices on the answer sheet (on the last page). If there is a problem with Moodle submission, please send this answer sheet to FLAS-exams@greenwich.ac.uk by the deadline.

Failure to follow any of these instructions may result in you failing the exam.

Answer all questions with the best answer(s):

1. Which of the following are operations in data structures (choose 3)?

   A. Traversal
   B. Subtraction
   C. Sorting
   D. Merging
   E. Travelling
   F. Debugging

   [4 marks]

2. Which of the following can be used explain what an algorithm is (choose 2)?

   A. A sequence of computational steps that transform input into the output.
   B. A method calls itself.
   C. A logical and mathematical model of a particular organisation of data.
   D. A loosely written code to make final code.
   E. A step-by-step procedure to solve a problem.

   [4 marks]

3. What is the Big-O performance of the following code (choose 1)?

```
7   def minmax(sequence):
8       size = len(sequence)
9       if (size == 0):
10          return (0,0)
11      min = max = sequence[0]
12      for val in sequence:
13          if (val > max):
14              max = val
15          if (val < min):
16              min = val
17      return (min,max)
```

   A. O(min)
   B. O(size)
   C. O(max)
   D. O(min+max)
   E. O(val)

   [4 marks]

4. … is very helpful in the situation when data items are stored and then retrieved in reverse order (choose 1 to fill in …)?

   A. Set
   B. Map
   C. Stack
   D. Queue
   E. None of the above

[4 marks]

5. What is the Big-O performance of the following code (choose 1)?

```
1.  def question(m, n):
2.      k = 0
3.      for i in range(m):
4.          for j in range(n):
5.              k += i * j
6.      return k
```

A. O(k)
B. O(m*k)
C. O(n*k)
D. O(m*n)
E. O(i*j)

[4 marks]

6. Consider the following code, what would be the output from lines 37, 39 and 40 (choose 1)?

```
1.  class ArrayStack:
2.
3.      def __init__(self):
4.          self.__stack = []
5.
6.      def len(self):
7.          return len(self.__stack)
8.
9.      def isEmpty(self):
10.         return self.len()==0
11.
12.     def push(self, value):
13.         self.__stack.append(value)
14.
15.     def view(self):
16.         return str(self.__stack)
17.
18.     def pop(self):
19.         if self.isEmpty():
20.             raise StackError('stack empty')
21.         else:
22.             return self.__stack.pop()
23.
24.     def peek(self, p = 0):
25.         if p>0 or (1 - p)>len(self.__stack):
26.             raise StackError('location beyond stack bottom')
27.         return self.__stack[p - 1]
28.
29.     def top(self):
30.         return self.peek()
31.
32.
33. if __name__ == '__main__':
34.     S = ArrayStack()
35.     S.push(5)
36.     S.push(3)
37.     print(S.pop())
38.     S.push(7)
39.     print(S.pop())
40.     print(S.pop())
```

A.
```
7
3
5
```
B.
```
3
5
7
```
C.
```
5
3
7
```
D.
```
3
7
5
```
E.
```
The code will compile but there will be an
exception raised.
```

[4 marks]

7. After adding 1, 3 and 7 into an empty queue, and then deleting one element at a time, in which order will they be deleted? (Choose 1)

A. 731
B. 137
C. 713
D. 317
E. None of the above

[4 marks]

8. When would it be ideal to use linear searching? (Choose 2)

A. When the list is linearly independent
B. When the list is sorted
C. When the list has only a few elements
D. When performing a few searches in an unordered list
E. When performing a search for duplicated values
F. Can be used all the time.

[4 marks]

9. What is the complexity order for the best case for a linear search of n items? (Choose 1)

A. O(1)
B. O(log n)
C. O(n)
D. O(n log n)
E. O(n^2)

[4 marks]

10. Which of the following would be a drawback when using linear search of n items? (Choose 1)

A. Requires more memory.
B. It is complex to understand.
C. Running time is more compared to other searching algorithms.
D. It requires looping implementation.
E. It cannot be programmed with recursion.

[4 marks]

11. Which of the following data structure CANNOT be stored in a linear type (Choose 1)

A. Stack
B. Queue
C. Binary Tree
D. List
E. None of the above

[4 marks]

12. Consider the following code for Bubble Sort, what would be the THIRD line of the output (from line 8)? (Choose 1)

```
1.  def bubbleSort(arr):
2.      n = len(arr)
3.
4.      # Traverse through all array elements
5.      for i in range(n):
6.
7.          # Last i elements are already in place
8.          print(arr)
9.
10.         for j in range(0, n-i-1):
11.
12.             # traverse the array from 0 to n-i-1
13.             # Swap if the element found is greater
14.             # than the next element
15.             if arr[j] > arr[j+1] :
16.                 arr[j], arr[j+1] = arr[j+1], arr[j]
17.
18. # Driver code to test above
19. arr = [64, 34, 25, 12, 22, 11, 90]
20.
21. bubbleSort(arr)
22.
23. print (arr)
```

A. [11, 12, 22, 25, 34, 64, 90]
B. [64, 34, 25, 12, 22, 11, 90]
C. [34, 25, 12, 22, 11, 64, 90]
D. [25, 12, 22, 11, 34, 64, 90]
E. There is an error with the code.

[4 marks]

13. What is the Big O performance of the worst case with an insertion sort of n items? (Choose 1)

A. O(n/2)
B. O(log n)
C. O(n)
D. O(n log n)
E. O(n^2)

[4 marks]

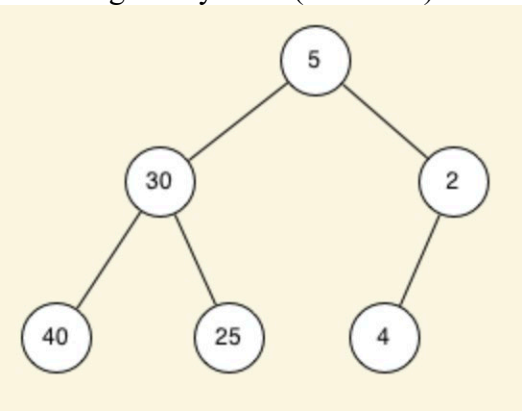
14. Consider the following code for Selection Sort, what would be the SECOND line of the output from line 16? (Choose 1)

```
1.  A = [64, 25, 12, 22, 11]
2.
3.  # Traverse through all array elements
4.  for i in range(len(A)):
5.
6.      # Find the minimum element in remaining
7.      # unsorted array
8.      min_idx = i
9.      for j in range(i+1, len(A)):
10.         if A[min_idx] > A[j]:
11.             min_idx = j
12.
13.     # Swap the found minimum element with
14.     # the first element
15.     A[i], A[min_idx] = A[min_idx], A[i]
16.     print(A)
```

A. [11, 12, 25, 22, 64]
B. [11, 12, 22, 25, 64]
C. [11, 25, 12, 22, 64]
D. [64, 25, 12, 22, 11]
E. There is an error with the code.

[4 marks]

15. What would be the pre-order traversal and post-order traversal of the following binary tree? (Choose 2)



A. 40, 25, 4, 30, 2, 5
B. 40, 25, 30, 4, 2, 5
C. 5, 30, 40, 25, 2, 4

D. 40, 30, 25, 5, 4, 2
E. 5, 30, 2, 40, 25, 4

[4 marks]

16. Consider the following code using Maps, what would be the output from line 2? (Choose 1)

```
1. m = {'a':1, 'b':2, 'd':3, 'c':3}
2. print(m['d'])
```

A. 0
B. 3
C. 4
D. None
E. There is an error with the code.

[4 marks]

17. Consider the following code using Maps, what would be the output at line 14? (Choose 1)

```
1. freq = {}
2. for char in "comp 1819 algorithms and data structure":
3.   w = ''.join(c for c in char if c.isalpha())
4.   if w:                                # require at least one alphabetic cha
   racter
5.     freq[w] = 1 + freq.get(w, 0)
6.
7. max_w = ''
8. max_c = 0
9. for (w,c) in freq.items():    # (key, value) tuples represent (w, c)
10.   if c > max_c:
11.     max_w = w
12.     max_c = c
13.
14. print(max_w, max_c)
```

A. a 4
B. t 4
C. comp 1
D. comp 4
E. There is an error with the given code.

[4 marks]

18. Consider the following code using Sets, what is the output at line 21? (Choose 1)

```
1. set1 = set([1, 2, 3, 4, 5, 6,
2.             7, 8, 9, 10])
3. print("Intial Set: ")
4. print(set1)
5.
6. # Removing elements from Set
7. set1.remove(1)
8. print(set1)
9.
10. # Discard elements from Set
11. set1.discard(3)
12. print(set1)
```

```
13.
14. # Removing elements from Set
15. for i in range(5, 10):
16.     set1.remove(i)
17. print(set1)
18.
19. set2 = set([2, 6])
20.
21. print(set1.union(set2))
```

A. {2, 4, 6, 8, 10}
B. {2, 4, 6}
C. {2, 10, 4, 6}
D. {2, 4, 10}
E. There is an error with the code.

[4 marks]

19. Identify the correct statements about a recursive method (choose 2)?

A. A method of solving a problem where the solution depends on solutions of smaller instances.
B. It can be an illegal call.
C. An object-oriented piece of code to execute an object.
D. A repeatedly executed block of code until a given condition is satisfied.
E. A method calls itself from its own code.

[4 marks]

20. What is the output of the following code (choose 1)?

```
7   def fibonacci(n):
8     """Return the nth Fibonacci number."""
9     if n <= 1:
10      return 3
11    else:
12      return fibonacci(n-2) + fibonacci(n-1)
13
14  print(fibonacci(3))
```

A. 1
B. 3
C. 5
D. 9
E. 11

[4 marks]

21. How many times are the method drawSpiral being called in the following code (choose 1)?

```
7   import turtle
8
9   myTurtle = turtle.Turtle()
10  myWin = turtle.Screen()
11
12  def drawSpiral(myTurtle, lineLen):
13      if lineLen > 0:
14          myTurtle.forward(lineLen)
15          myTurtle.right(90)
16          drawSpiral(myTurtle,lineLen-10)
17
18  drawSpiral(myTurtle,35)
19  myWin.exitonclick()
20
```

A. 1
B. 4
C. 2
D. 5
E. None of the above.

[4 marks]

22. What is the Big-O performance for merging two sorted lists of size p and q (choose 1)?

A. O(p|q)
B. O(p+q)
C. O(p*log q)
D. O(q*log p)
E. O(p*q)

[4 marks]

23. Given an already-sorted array, identify which sorting algorithms would have the best running time? (Choose 2)

A. Bubble Sort with a swap flag
B. Insertion Sort
C. Selection Sort
D. Quick Sort
E. Merge Sort

[4 marks]

24. What is the output of the following code (choose 1)?

```
def inplace_quick_sort(S, a, b):
  """Sort the list from S[a] to S[b] inclusive using the quick-sort algorithm."""
  if a >= b: return                                  # range is trivially sorted
  pivot = S[b]                                       # last element of range is pivot
  left = a                                           # will scan rightward
  right = b-1                                        # will scan leftward
  while left <= right:
    # scan until reaching value equal or larger than pivot (or right marker)
    while left <= right and S[left] < pivot:
      left += 1
    # scan until reaching value equal or smaller than pivot (or left marker)
    while left <= right and pivot < S[right]:
      right -= 1
    if left <= right:                                # scans did not strictly cross
      S[left], S[right] = S[right], S[left]          # swap values
      left, right = left + 1, right - 1              # shrink range

  # put pivot into its final place (currently marked by left index)
  S[left], S[b] = S[b], S[left]
  # make recursive calls
  inplace_quick_sort(S, a, left - 1)
  inplace_quick_sort(S, left + 1, b)

S = [85, 24, 63, 45, 17, 31, 96, 50]
inplace_quick_sort(S, 0, 6)
print(S)
```

A. [85, 17, 24, 31, 45, 50, 63, 96]
B. [85, 17, 24, 31, 45, 63, 96, 50]
C. [17, 24, 31, 45, 63, 85, 96, 50]
D. [17, 24, 31, 45, 85, 63, 96, 50]
E. [17, 24, 31, 45, 50, 63, 85, 96]

[4 marks]

25. How many times are the method merge_sort being called in the following code (choose 1)?

```python
def merge(S1, S2, S):
  """Merge two sorted Python lists S1 and S2 into properly sized list S."""
  i = j = 0
  while i + j < len(S):
    if j == len(S2) or (i < len(S1) and S1[i] < S2[j]):
      S[i+j] = S1[i]      # copy ith element of S1 as next item of S
      i += 1
    else:
      S[i+j] = S2[j]      # copy jth element of S2 as next item of S
      j += 1

def merge_sort(S):
  """Sort the elements of Python list S using the merge-sort algorithm."""
  n = len(S)
  if n < 2:
    return                # list is already sorted
  # divide
  mid = n // 2
  S1 = S[0:mid]           # copy of first half
  S2 = S[mid:n]           # copy of second half
  # conquer (with recursion)
  merge_sort(S1)          # sort copy of first half
  merge_sort(S2)          # sort copy of second half
  # merge results
  merge(S1, S2, S)        # merge sorted halves back into S

S = [85, 24, 63, 45, 17, 31, 96, 50]

merge_sort(S)

print(S)
```

A. 15
B. 7
C. 3
D. 1
E. None of the above.

[4 marks]

Answer Sheet

TITLE OF PAPER Algorithms and Data Structures COURSE CODE COMP1819

Your Student ID (e.g 000123456): 00_____

Please circle all correct answers

1.	A	B	C	D	E	F
2.	A	B	C	D	E	
3.	A	B	C	D	E	
4.	A	B	C	D	E	
5.	A	B	C	D	E	
6.	A	B	C	D	E	
7.	A	B	C	D	E	
8.	A	B	C	D	E	F
9.	A	B	C	D	E	
10.	A	B	C	D	E	
11.	A	B	C	D	E	
12.	A	B	C	D	E	
13.	A	B	C	D	E	
14.	A	B	C	D	E	
15.	A	B	C	D	E	
16.	A	B	C	D	E	
17.	A	B	C	D	E	
18.	A	B	C	D	E	
19.	A	B	C	D	E	
20.	A	B	C	D	E	
21.	A	B	C	D	E	
22.	A	B	C	D	E	
23.	A	B	C	D	E	
24.	A	B	C	D	E	
25.	A	B	C	D	E	