Important Information

Any **algorithms** you produce should be **written in words**, you will be told to write a python program or function if you are to write in python.

All Python code you write for this exam must satisfy the following requirements:

- Syntax should satisfy Python 3 requirements
- Use syntax, functions, structures and constructs presented in lectures.
- avoid using any inbuilt python functions/methods which make any tasks significantly simpler

Write down any assumptions you make.

Do not write anything in this table. It is for office use only.

Question	Points	Score
1	4	
2	4	
3	6	
4	5	
5	6	
6	6	
7	6	
8	6	
9	9	
10	10	
11	8	
Total:	70	

0

Circle one letter for each part corresponding to the correct response

If you change your mind, clearly cross out your previous choice and circle the new one and write the letter chosen on the side

Question 1: [4 marks]

(a) (1 mark) what will be printed by the following code?

```
initial = 123
pos = 0
s=0
while pos < len(str(initial)):
        v = str(initial)[pos]
        s = s + int(v)
        pos = pos + 1
print(s)
```

A. 6

B. 3

C. 123

D. 0

E. nothing; an error would occur or the code would not terminate

(b) (1 mark) what will be printed by the following code?

```
aList = ["1","2"]
bList = ["3","2"]
cList = aList + bList
value = ""
for index in range(len(cList)):
        value = cList[index] * index
print(value)
```

A. ""

B. "222"

C. "33"

D. "233222"

E. nothing; an error would occur or the code would not terminate

(c) (1 mark) what will be printed by the following code?

```
aString = "my string is too short"
while len(aString) > 10:
        aString = aString + "!"
print(len(aString))
```

A. 0

B. 10

C. 11

D. 22

E. nothing; an error would occur or the code would not terminate

(d) (1 mark) What will be printed after the following code is run?

```
def whatIsThis(aList,p1,p2):
        tmp = aList[p1]
        aList[p1] = aList[p2]
        aList[p2] = tmp


myList = [10,20,30,40]
whatIsThis(myList,1,3)
print(myList)
```

A. [10,20,30,40]

B. [10,20,30,20]

C. [30,20,10,40]

D. [10,40,30,20]

E. nothing; an error would occur or the code would not terminate

2

Question 2: [4 marks]

For each of the following, list the complexity and explain why it has this complexity.

(a) (2 marks) assume N is len(bList)

```
def bFunc(bList):
        b = 1
        k = 0
        while k < len(bList):
                b = b * bList[k]
                k = k + 2
        return b
```

O(N) - k changes by 2 at a time meaning this will run for N/2 iterations (with each iteration constant) but constant factors are ignored meaning O(n) is correct

- 1 mark for a correct complexity
- 1 mark for valid justification

working out from the RAM model is fine

(b) (2 marks) assume N is len(cList)

```
def cFunc(cList):
        c = 1
        k = 0
        half = len(cList)//2
        while k < len(cList):
                c = c - cList[k]
                k = k + half
        return c
```

O(1) - k changes by n/2 at a time meaning this will run for 2 iterations (with each iteration constant) regardless of the size of n

- 1 mark for a correct complexity
- 1 mark for valid justification

working out from the RAM model is fine

4

Question 3: [6 marks]

This question is about stacks and queues.

(a) (3 marks) Consider a stack as below:

```
bottom -> [5, 20, 3] <- top
```

Assuming that the right end of the above list represents the top of the stack, draw the state of the stack after each of the following operations are run on it (0.5 marks per correct state).

Be sure to make clear what the top of the stack is in each diagram.

1. push 5
2. push 10
3. pop
4. pop
5. pop
6. push -5

```
bottom -> [5, 20, 3, 5] <- top
bottom -> [5, 20, 3, 5, 10] <- top
bottom -> [5, 20, 3, 5] <- top
bottom -> [5, 20, 3] <- top
bottom -> [5, 20] <- top
bottom -> [5, 20, -5] <- top
```

- 0.5 marks per correct state (based on the previous state)

(b) (3 marks) For the following program, show the state of the queue Q after each iteration of the loop. You may assume the following:

- append(aQueue,A) - will append A into the queue aQueue
- serve(aQueue) - will serve from the queue aQueue
- size(aQueue) - will return the number of items in the queue aQueue
- the left end of Q is the front and the right end of Q is rear

```
def maxInQueue(aQueue):
        N = size(aQueue)
        theMax = 0
        while N > 0:
                item = serve(aQueue)
                if item > theMax:
                        theMax = 0.5*item
                        append(aQueue,theMax)
                N -= 1
        return theMax

Q = [1,15,6,8,3,1]
print(maxInQueue(Q))
```

```
[1,15,6,8,3,1] - start state (not marked)
[15,6,8,3,1,0.5]
[6,8,3,1,0.5,7.5]
[8,3,1,0.5,7.5]
[3,1,0.5,7.5,4]
[1,0.5,7.5,4]
[0.5,7.5,4]
```

- 0.5 marks per correct state (based on previous state)

3

Question 4: [5 marks]

This question is about sorting and algorithms.

(a) (1 mark) Given a list $L = [1, 6, 1, 6, 9, 2, 4, 2, 6, 1, 2, 1]$ show what list counting sort would create in order to sort this list. Be sure to specify what the indices and values represent.

```
[0,4,3,0,1,0,3,0,0,1]
```

- the index is a number from 0 to 9
- the value is how many times that number occurred in L

- 0.5 marks indices clearly representing possible elements from L
- 0.5 marks values representing frequency of that element in L

(b) (4 marks) Give an **algorithm** which describes how selection sort sorts a list of numbers

```
1. input the list to sort
2. the first position P is where to swap to
3. find the smallest from P to the end
4. swap this smallest with position P
5. repeat until P has reached the end of the list
6. the list is now sorted; output sorted list
```

- 1 mark for describing the finding the max/min
- 1 mark for describing the swapping elements
- 1 mark for describing how to move onwards
- 1 mark for determining when to terminate

5

Question 5: [6 marks]

Select an algorithm (such as the algorithm you wrote for selection sort earlier) and **explain** why it is an algorithm. Be sure to consider each of the properties of algorithms (that we discussed in lectures) in your answer.

Students could plausibly choose selection sort, insertion sort, counting sort, quicksort, merge sort or heap sort.
Students should have listed each of the following

- finiteness
- sequence of steps
- definiteness
- preciseness
- input ($>=0$)
- output ($>=1$)

they should then explain how each of these points is met by the algorithm of their choice
for each attribute of algorithms

- 0.5 marks for listing
- 0.5 marks for explaining how the chosen algorithm meets that attribute

6

Question 6: [6 marks]

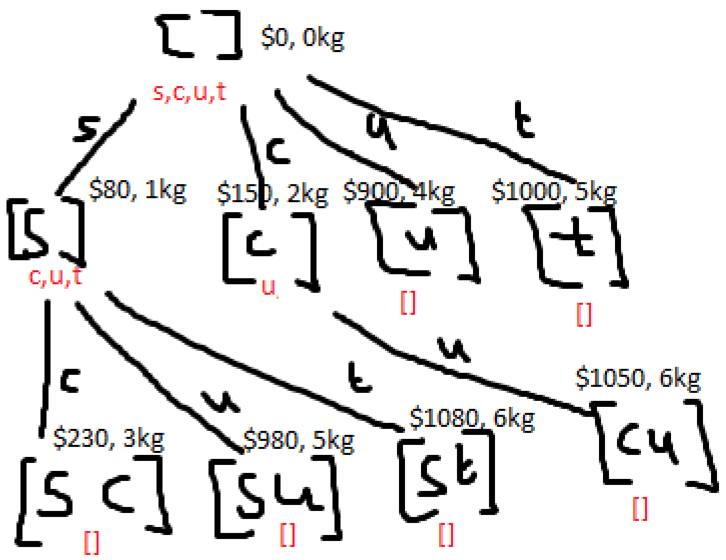
Consider the knapsack problem for a maximum weight of 6kg for the following set of items.

Table 1: items available for knapsack

Item	Spoon	Candlestick	Urn	Television
Weight	1	2	4	5
Value	80	150	900	1000

Assuming this problem was solved using the backtracking method, show a back-tracking diagram for this problem. This should clearly show the partial solutions, next options and complete solutions.

For convenience, you may shorten the names of spoon, candlestick, urn and television to s, c, u and t respectively.



above in red (beneath each state) we have the next available states
each state has square brackets around it and letters to signify which elements are part of this
the cost and weight of each state is shown above each state.
the item chosen is shown along the lines from one state to another.

- 0.5 marks for the initial empty state
- 0.5 marks for each of the other partial and complete solutions (4 marks total)
- 0.5 marks for each of the next states where non-empty (1.5 marks total)

6

Question 7: [6 marks]

This question is about the transform and conquer strategy.

Given the maxheap represented as an array as [7,4,1,0,3,1]. Show what happens to the heap by performing the extract maximum method **twice**. Ensure you show the state of the heap (in either array or tree representation) after every swap. If at any point no swap occurs, explain why.

Reminder: the children of any node in the array representation can be found at $2 \times i$ *and* $1 + 2 \times i$ *(where indexing begins at 1)*

```
[7,4,1,0,3,1] //start state
//extract max
->[1,4,1,0,3,_7_] //root and last item swap (_ to show 7 no longer in heap)
-> [4,1,1,0,3] //1 swaps with larger child (4)
-> [4,3,1,0,1] //1 swaps with larger child (3)
//7 has been extracted
//extract max
-> [1,3,1,0,_4_] //root and last item swap (_ to show 4 no longer in heap)
-> [3,1,1,0] //1 swaps with largest child (3)
-> [3,1,1,0] //1 remains where it is since the only child (0) is smaller
```

- 1 mark per correct state (based on previous)

in the case of the very last state (no swap) there should be an explanation that the child of 1 is already smaller so no need to swap further down

6

Question 8: [6 marks]

Give a useful invariant for the inner loop in terms of j and k and the outer loop in terms of k and list what this code does. [2+2+2=6]

```
def cFunc(cList):
        k=1
        while k < len(cList):
                j = k-1
                while j >= 0:
                        if cList[j] > cList[j+1]:
                                tmp = cList[j]
                                cList[j] = cList[j+1]
                                cList[j+1] = tmp
                        j = j - 1
                k = k + 1
        return cList
```

at the end of the jth iteration, the last j items (up until position k) are sorted (eg. at iteration j, positions k-j...k are sorted)
hence after all j operations are complete the first k items are in order
at the end of the kth iteration, the first k items are sorted and hence after the nth iteration all of the items are sorted so this is a sorting algorithm (in fact it is insertion sort).

- 2 marks for each correct, useful invariant
  - 1 mark if invariant is vague
- 1 mark for recognising the code does some kind of ordering
- 1 mark if they recognise this as insertion sort (even if they don't name it as such)

6

Question 9: [9 marks]

This question is about recursion and divide and conquer approaches to problem solving. Consider the task of counting instances of a particular character in a string S.

(a) (6 marks) Consider the recursive definition given below:

$$Count(i, S, target) = Count(i-1, S, target) + \begin{cases} 1 \text{ where } S[i] \text{ is target} \\ 0 \text{ where } S[i] \text{ is not target} \end{cases}$$

Note that $Count(i, S, target) = 0$ *where* $i < 0$

Eg. consider the string $S =$`"ab cbb ef b gh ijkl"` with a target of `"b"`. Here:

$count(0, S, target) = 0$ as position 0 is not b

$count(1, S, target) = 1$ as position 1 is b

$count(2, S, target) = 1$ as position 2 is not b

$count(3, S, target) = 1$ as position 3 is not b

$count(4, S, target) = 2$ as position 4 is b

$count(5, S, target) = 3$ as position 5 is b

write a **recursive python function** *Count* which accepts as input the position, i, string, S, and target, *target*, and implements this relationship, returning the result. *Note: you do not need to consider the case of running count with an i value exceeding the number of characters in the string*

```
def count(i,S,target):
        if i<0:
                return 0
        else:
                if S[i]==target:
                        return 1 + count(i-1,S,target)
                else:
                        return count(i-1,S,target)
```

- 0.5 marks for correct function definition
- 1 mark for correct base case (i of 0 also valid)
- 1 marks for checking if there is a match
- 1 marks for correctly combining solutions
- 2 mark for correctly calling count recursively
- 0.5 marks for returning result

6

(b) (3 marks) Describe in words how the divide and conquer approach can be applied to this problem.

Divide and conquer can be applied here by cutting the string in half repeatedly until there is only one element to consider, if this element is the target that section evaluates to a 1 otherwise to a 0 and we combine this result with the other half by addition.

- 1 mark for clarifying how the problem is solved with 1 (or 0) len strings
- 1 mark for explaining the divide stage
- 1 mark for explaining the combine stage

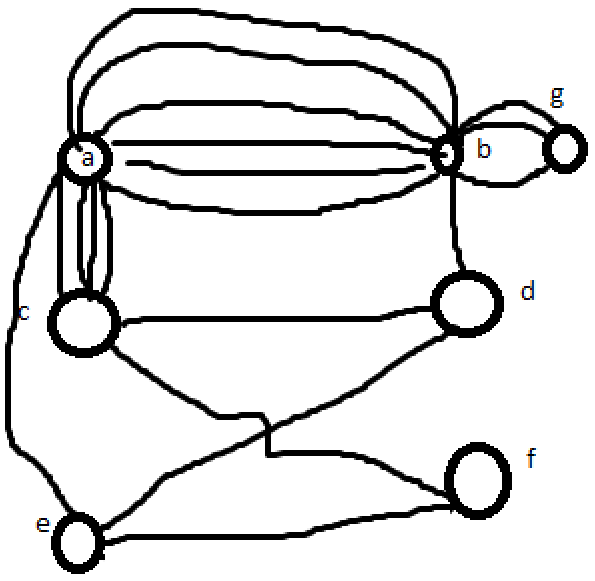
3

Question 10: [10 marks]

Consider the problem of finding the smallest vertex cover for a given graph G. This question has you considering a greedy approach and contrasting with a brute force solution.

(a) (7 marks) Write an **algorithm** (*greedyCover*) which accepts a graph (G) and, using the principle of greed, finds a set of vertices corresponding to the smallest vertex cover (or something close to it). Your algorithm should output the vertices in the vertex cover found.
Note: Ensure you clarify where your algorithm is applying greed. If you are unsure how to write this as an algorithm, you can still get up to 3 marks by explaining how you were intending to apply greed in this situation
*Note2: Given a greedy approach will not always give an optimal solution to this problem, your algorithm is not expected to consistently give the **best** answer but it should still provide a vertex cover and demonstrate understanding of greed.*

In considering their answer you might like to compare it against the following graph:



They will likely produce a function which steps through the adjacency list and picks those with the largest degree first. One possible algorithm below:

```
1. initialise a list (cover) to empty
2. repeat until no edges remain in the graph
    (a) for each vertex in the graph
        i. find the vertex of greatest degree
        ii. add this vertex to cover
        iii. remove all edges attached to this vertex from the graph (hence
             reducing this vertex's degree so it is not rechosen)
3. output the list cover
```

7

- 1 mark for stopping when all edges are covered by the vertex cover
- 0.5 mark for considering each vertex as a candidate
- 0.5 marks for avoiding selecting the same vertex more than once
- making a greedy choice about a given vertex:
  - 1 mark if they are able to choose between vertices using this strategy
  - 2 mark if the strategy is optimal at the time
    - 1 mark if the strategy only **sometimes** selects an optimal vertex at the given time (eg. based on degree in the original graph but without updating as vertices are added to the cover)
- 1 mark for keeping track of vertices in the cover
- 1 mark for somehow keeping track of remaining edges (ex. in my code we've just removed used edges)

(b) (3 marks) Explain in words what a brute force solution to this problem would entail.

there are three aspects to this:

- generate subsets to represent inclusion or exclusion of every vertex from the vertex cover (bitlist)
- determine which of these subsets cover every edge (ie. which subsets are vertex covers)
- output the smallest size subset which is a vertex cover

- 1 mark for each of the three aspects responded to

3

Question 11: [8 marks]

Consider a variant of the N-Queens problem which aims to find a solution to N-Queens which also satisfies the restriction that if you add together the row positions of the queens in the first two columns on the board, they add to some value k. For example, the 6 queens problem and a k value of 7, there is just one solution (of the four solutions to 6 Queens) which meets this restriction.

(a) (1 mark) Explain what a certificate to this problem must include.

- a set of coordinates for each queen (this could be a list of [row,col] lists, or numbered cells, or an ordered list of row positions, etc.)

(b) (5 marks) Explain in words how you would check the certificate for this problem for a k value of 10.

```
1. if numQueens in certificate $\neq N$ return False
2. for each queen...
   (a) for each other queen...
       i. if pair of queens can attack horizontally return False
       ii. if pair of queens can attack vertically return False
       iii. if pair of queens can attack diagonally return False
   (b) if queen is out of bounds return False
3. add row num of first two column's queens together...
4. if sum = k return True
5. else return False
```

- 0.5m certificate is of correct length
- 0.5m each item represents a valid board position
- 0.5m considers each queen position
- 0.5m considers each other queen
- 1m checks for column,row attacks (0.5 each)
- 1m checks for diagonal attacks
- 0.5m get sum
- 0.5m sum==k

6

(c) (2 marks) Using **only** your answer to the previous part, argue as to whether this variant of the N-Queens problem is a P or NP class of problem. Justify your answer.

Their previous algorithm should be in the P class, which means that the problem is at worst an NP problem. The definition of an NP problem is one which can be verified in P time, given we have just constructed an algorithm which verifies a certificate for the problem and completes in P time, the problem is hence an NP problem. It may also be the case that it is a P problem however we do not have enough information to make that argument.
note, strictly speaking this problem can be solved in P time however if they are answering **purely** from their answer to the previous part they can neither confirm nor deny that it can be solved in P time, just that they know it can be checked in P time

- 0.5 marks for saying NP
- 1 mark for explaining verification in P makes something NP
- 0.5 marks for noting it may still actually be P

2