EXAMPLE TCA. Note this is **not** a past paper **

Coventry University

Faculty of Engineering, Environment and Computing

5008CEM

Programming For Developers

Instructions to candidates

The total number of questions in this paper: 4

All questions carry equal marks

Start each question **on a new page** and carefully identify your answers with the correct question number

For this examination you will be supplied with the following: 1 Answer Book

You must may take this question paper away at the end of the examination
Please keep in a safe place for future reference

Question 1 Total for Question 1: **25 Marks**

a) Use pseudocode to describe an algorithm that counts all occurrences of a particular character in a string.

Input: a string s, for example s = 'A beautiful day!'

a character c, for example c = 'a'

Output: occurrences count nr = 3

To solve this task, consider the string to be an array of characters and do not use any predefined functions that would make this task trivial. (10 marks)

b) Implement the algorithm in Python. You will not be penalised for small syntactical errors. (7 marks)

c) Implement an algorithm in Python that reverses a string recursively.

Input: a string s read from the keyboard, for example s = 'beautiful'

Output: a string s = 'lufituaeb'

To solve this task, consider the string to be an array of characters and do not use any predefined functions that would make this task trivial. (8 marks)

(continue)

Question 2 Total for Question 2: **25 Marks**

a) Perform a manual Bubble Sort to rearrange the following numbers into ascending order. State the number of comparisons and the number of swaps for each of the first three passes.

13 16 10 11 4 12 6 7

(9 marks)

b) Name the best sorting algorithm for each of the following scenarios. Select from bubble sort, heap sort, insertion sort, merge sort, selection sort, or quick sort.
1 The array is almost entirely sorted in its original form.
2 You cannot use extra space (except perhaps a couple of local variables) and its worst case performance has to be O(nlogn).
3 You have many small data sets to sort separately (each with less than 10 elements). (6 marks)

c) Use Big-O notation to describe the time complexity of Quick Sort, relative to input size. (4 marks)

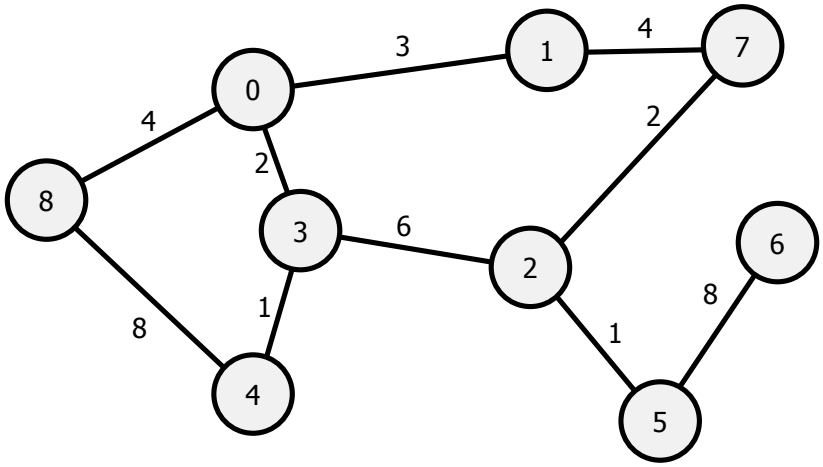
d) Complete this Python code, showing how you would use `datetime` to measure the program's time performance. Report the total number of solutions and the time taken with information e.g. 'the total number of solutions is <no of solutions>'; 'the total time taken was <total time taken>'. The necessary imports, and the first line of the solution, have been included. (6 marks)

```
""" Solves the '8 queens' problem for n queens, producing all
    solutions and returning the total number of solutions. """

from datetime import datetime

def is_available(n, table, column, N):
    return not any(t in (n, n - i, n + i) for t, \
                   i in zip(table, range(column, 0, -1)))

def queens_sum(N):
    return solve(N, [0]*N, 0, N)

def solve(N, table, column, end):
    if column == end:
        print('table is: ', table)
        return 1
    summ = 0
    for n in range(N):
        if is_available(n, table, column, N):
            table[column] = n
            summ = summ + solve(N, table, column+1, end)
    return summ

start = datetime.now()
```

(continue)

Question 3 Total for Question 3: **25 Marks**

a) Perform Prim's algorithm manually using the graph below. Start with 0. Use the table below the graph to show your solution. Add lines as necessary. The first line has been done for you. (8 marks)



Edge	Weight
0 - 3	2

(This question continues on the next page)

b) Complete this Python implementation of Prim's algorithm by implementing the minKey function. Comments are provided to support your work. Write the code on a separate page if you wish. Minor syntax errors will be ignored. (17 marks)

```
import sys                                                  #needed for maxsize

class Graph():

    def __init__(self, vertices):                          #implements graph as adjacency matrix
        self.V = vertices                                  #number of vertices
        self.graph = [[0 for column in range(vertices)]    #adjacency matrix with no edges (all connections set to zero)
                    for row in range(vertices)]

    def printMST(self, parent):
        print ("Edge \t Weight")
        for i in range(1, self.V):
            print (parent[i], "-", i, "\t", self.graph[i][ parent[i] ])

"""from reached nodes find the unreached node with the minimum cost"""
    def minKey(self, key, mstSet):
                                                           #set min to infinity (use maxsize which is next best thing!)
                                                           #count through number of vertices
                                                           #if vertex is less than min and unreached
                                                           #assign to min
                                                           #min_index is position of min in key
                                                           #return min_index

"""find MST"""
    def primMST(self):
        key = [sys.maxsize] * self.V                       #list of values all set to infinity
        parent = [None] * self.V                           #list for constructed MST
        key[0] = 0                                         #set first element of key to zero
        mstSet = [False] * self.V                          #mstSet is list of booleans set to False
        parent[0] = -1                                     #first element of parent list set to -1
        for vertex in range(self.V):                       #go through all vertices
            u = self.minKey(key, mstSet)                   #call minKey; minKey returns u (index of unreached node)
            mstSet[u] = True                               #mstSet at index of node is set to True
            for v in range(self.V):                        #go through all vertices, choose lowest cost path

                if self.graph[u][v] > 0 and mstSet[v] == False and key[v] > self.graph[u][v]:
                    key[v] = self.graph[u][v]
                    parent[v] = u                          #parent[v] is index of node; list of parents (nodes) is the MST
        self.printMST(parent)                              #print the list of parents, i.e. the MST
```

(This question continues on the next page)

```
g = Graph(5)
g.graph = [[0, 2, 0, 6, 0],
           [2, 0, 3, 8, 5],
           [0, 3, 0, 0, 7],
           [6, 8, 0, 0, 9],
           [0, 5, 7, 9, 0]]

g.primMST()
```

(continue)

Question 4 Total for Question 4: 25 Marks

a) What error could this Python code produce, and why? (10 marks)

```
from threading import Thread, Event
from time import time
from time import sleep

event = Event()

def count_up_in_twos(var):
    while True:
        for i in range(len(var)):
            var[i] += 2
        if event.is_set():
            break
    print('Stop printing')

twos_var = [1, 2, 3]
t = Thread(target=count_up_in_twos, args=(twos_var, ))
t2 = Thread(target=count_up_in_twos, args=(twos_var, ))
t.start()
t2.start()
t0 = time()
while time()-t0 < 5:
    try:
        print(twos_var)
        sleep(1)
    except KeyboardInterrupt:
        event.set()
        break
event.set()
t.join()
t2.join()
print(twos_var)
```

b) Rewrite the code to prevent the error, using a queue or queues. Explain your solution, including your decision on the number of queues. (15 marks)

End