1. Write the time complexity of the following code snippet. Show your works/reasoning. **(3*2=6 marks)**

a)
```
for (i =1; i*i <= n; i++)   ⟶ √n ⟵ ( i² <= n
                                      ∴ i <= √n )
   for (j =1; j<= n; j++)   ⟶ n                   } O(n.√n)
      sum += i + j
```

b)
```
worstCase(n):
   for (int  i = 1; i < n; i = i * 3)              ⟶ log₃ n
      for (int j = n; j >= 1; j = j / 5)           ⟶ log₅ n        } O(n.log₃n.log₅n)
         for (int  k = 1; k <= n; k = k + 5)       ⟶ n/5           } Or, O(n(log n)²)
            sum += i + j + k
```

c)
```
worstCase(n):
   for (int  i = 0; i < n; i = i * 3)              ⟶ i=0, will always
      for (int j = n; j >= 1; j = j / 5)              remain 0
         for (int  k = 1; k <= n; k = k + 5)        ∴ infinite loop
            sum += i + j + k
```

1. Write the time complexity of the following code snippet. Show your works/reasoning. **(3*2=6 marks)**

line 4 executes the most

a)
```
1.  for (k=1; k<=n; k++)
2.      j=1;
3.      while (j <= k*k)
4.          j= j+1;
```

$\rightarrow$ # times line 4 executes

$k=1$ … $1 = 1^2$

$k=2$ … $4 = 2^2$

$k=3$ … $9 = 3^2$

$\vdots$

$k=n$ … n^2

$$1^2+2^2+\dots n^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$$

b)
```
worstCase(n):
   for (int  i = 0; i < n; i = i + 3)
      for (int j = n; j >= 1; j = j / 5)
         for (int  k = 1; k <= n; k = k * 5)
            sum += i + j + k
```

$\longrightarrow n/3$

$\longrightarrow \log_5 n$

$\longrightarrow \log_5 n$

$$O(n \cdot \log_5 n \cdot \log_3 n)$$

2. For each of these pairs of functions (f(n), g(n)), identify whether f(n) = O(g(n)) or f(n) = □(g(n)). Write “True” or “False” for the two options in each case in the question paper.
(10 marks)

f(n)	g(n)	f(n) = O(g(n))	f(n) = Ω(g(n))
3^n	$2^n.n^4$	F	T
$(\sqrt{n} + n)(30\sqrt{n})$	n^2	T	F
$n^2 \log n^n$	n^3	F	T
$\log n + 5$	$\log n^2$	T	T
$\log n$	$\log \log n$	F	T

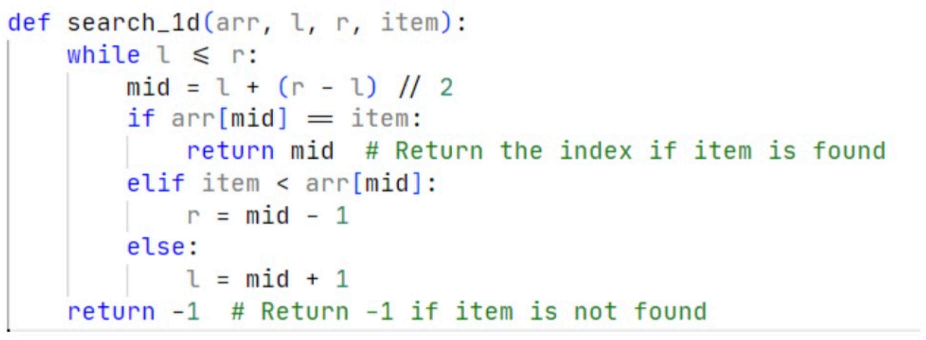
3. You are studying a searching algorithm used to find a specific value in a sorted array. This algorithm operates by dividing the search interval in half at each step, comparing the target value to the midpoint value, and then discarding one half of the search interval based on the comparison, continuing with the remaining half.
a. Based on this description, identify the algorithm used. **(1 mark)**
b. Consider you are given a rotated sorted array (e.g [30, 40, 50, 60, 70, 80, 90, 100, 10, 20]). Now, propose a modified version of the described algorithm to find whether a value exists in the rotated sorted array or not.. **(3 marks)**

2. For each of these pairs of functions (f(n), g(n)), identify whether f(n) = O(g(n)) or f(n) = □(g(n)). Write "True" or "False" for the two options in each case in the question paper. **10 marks**

f(n)	g(n)	f(n) = O(g(n))	f(n) = □(g(n))
$\log n^2$	$\log n + 5$	T	T
$(\sqrt[3]{n} + n)(30\sqrt{n})$	$\sqrt{n}$	F	T
n^3	$n^2 \log n^n$	T	F
$3^n.n^4$	2^n	F	T
$\log \log n$	$\log n$	T	F

3. Consider an 2D sorted array with m rows and n columns. Your task is to check if an item is in the 2D array or not. Now, write an efficient pseudocode or program with logarithmic time complexity to find the row and column of an item in this 2D array. If the item is not in the 2D array the program should print -1. To help you solve this problem, a python implementation of searching in 1D array is given which you can use in your code. You can call this function anywhere in your program

4



```python
def search_1d(arr, l, r, item):
    while l <= r:
        mid = l + (r - l) // 2
        if arr[mid] == item:
            return mid  # Return the index if item is found
        elif item < arr[mid]:
            r = mid - 1
        else:
            l = mid + 1
    return -1  # Return -1 if item is not found
```