Key

RV UNIVERSITY Go, change the world an initiative of RV EDUCATIONAL INSTITUTIONS School of Computer Science & Engineering B.Tech(H) Program	**Class Test #1 - set-1** **Academic Year: 2023-24** **Term: Aug 02 to Nov 29, 2023** **Semester: 3 Section: A,C**
	Date: **Time:** **USN:** **Student Name:** **Course Code:** CS2000 **Course Name:** Design & Analysis of Algorithms **Max Marks:** 20 (scaled to 10)

Using/watching mobile Phones, smart watches or any other internet enabled devices are treated as malpractice.

Instructions to Students:

1. ***Answer all questions.***
2. ***You have to implement program for Q# 3.a after handing over the paper & upload in Google Classroom.***
3. ***Answers for all the questions need to be written on paper (except the program (Q3 3.a) which has to be implemented on hackerrank and later the source file has to be submitted on Google classroom).***

1. a) Write pseudocode/algorithm for finding median of array elements **(2 marks)**

```
Algorithm findMedian(A[0,1,...n-1])
{
    sort(A[0,1,...n-1])
    if n%2==0
        return (A[(n-1)/2] + A[n/2])/2
    else
        return A[n/2]
}
```

```
Algorithm sort(A[0,1,...n-1])
{
    for i=0 to n-2 do
        for j=0 to n-2-i do
            if A[j+1] < A[j]
                swap(A[j+1], A[j
}
```

b) What is/are the basic operation(s) of this algorithm? **(1 mark)**

Comparisons, swappings.

e) What is the time complexity of this algorithm? **(2 marks)**

$\Theta(n^2)$ if bubble sort, selection sort used for sorting.

f) What is the Best Case, Average Case & Worst Case Efficiency? **(1 mark)**

* Blooms Level - **R:** Remember **U:** Understand **Ap:** Apply **An:** Analyze **E:** Evaluate **C:** Create

$\Theta(n^2)$

2. **Given the below algorithm,**

```
Alg(n)
{
    int i, j, k, n;
    for(i=n/2;i<=n;i++)
    {
        for(j=1;j^2<=n/2;j++)
        {
            for(k=1;k<=n;k=k*2)
                print("DAA")
        }
    }
}
```

$j <= \sqrt{n/2}$

a. compute the time complexity of the above algorithm **(3 marks)**

$i = n/2$ The two outer loop variables are not dependent on each other. and the loop variable is incrementing linearly.

Inner loop k value is incremented in power's of 2, $2^0, 2^1, 2^2 \ldots 2^p$

if $2^p > n$, then the inner loop terminates. $2^p > n \Rightarrow p = \log_2 n$

So total time complexity is $\Theta(n\sqrt{n}\log_2 n)$

b. What is the time complexity of above algorithm if i is increased by, i = i*3 (i times 3, not i power 3) **(1 mark)**

$\Theta(n\sqrt{n}\log_3 n)$

c. Given f(n) = 100 n^3+10 n^2, g(n) = n^3, verify f(n) = O(g(n)) is true or not? **(2 marks)** True

To verify, $f(n) \stackrel{?}{=} O(g(n))$, we need to show that

$f(n) \le c \cdot g(n)$, $100n^3 + 10n^2 \le c \cdot n^3$ such that $c > 0, n_0 \ge 1$

for $c = 111$, $n_0 \ge 1$ always

$100n^3 + 10n^2 = O(n^3)$ $f(n) \le c \cdot g(n)$

d. What is the time complexity of an algorithm that does not have either iteration or recursion in it? **(1 mark)**

* Blooms Level - **R:** Remember **U:** Understand **Ap:** Apply **An:** Analyze **E:** Evaluate **C:** Create

$O(1)$ or $O(c)$ constant amount of time.

3.

a. Implement an algorithm that returns 1 if the array has unique elements otherwise it returns -1. Hackerrank contest will have a detailed problem description for this. **(3 marks)**

```
for i = 0 to n-2 do
    for j = i+1 to n-1 do
        if A[i]=A[j] return -1
return 1
```

b. What is the time complexity of the algorithm for which you have written the code. **(2 marks)**

~~Assumi~~ If unsorted array is taken directly to find unique elements. Time complexity of that algorithm is $O(n^2)$

c. Can you think of a better algorithm (in terms of time efficiency) compared to yours? **(2 marks)**

~~If~~ If the unsorted array is sorted first, before finding unique elements. then ~~the~~ finding unique elements can be performed in linear time. and sorting ~~requires~~ takes (merge sort, quick sort) $O(n \log n)$

So over all, time complexity is ~~O(~~ $n\log n + n$

$O(n\log n)$

* Blooms Level - **R:** Remember **U:** Understand **Ap:** Apply **An:** Analyze **E:** Evaluate **C:** Create