Roll Number:

Thapar University, Patiala

Department of Computer Science and Engineering

BE (Electronics and Computer Engineering) Semester – V
Date: September 23, 2016

Course Code: UCS406; Course Name: Data Structures and Algorithms
Time: 2 Hours, M. Marks: 30 (To be normalized)
Name of Faculty: RKS

Note: Attempt all the problems. Please attempt the parts of a problem at one place. You can assume a missing data suitably. You can use the following, wherever required:

$\sum_{k=1}^{n} \frac{1}{k} = \log n + O(1); \sum_{k=1}^{n} k^2 = n(n+1)(2n+1)/6$

1(a)	Assume that you are given an array $A[1 \ldots n]$ of distinct numbers. This is also known that the sequence of numbers in the array is unimodal, *i.e.*, there is an index i such that the sequence $A[1 \ldots i]$ is increasing ($A[j] < A[j+1]$ for $1 \leq j < i$), and the sequence $A[i \ldots n]$ is decreasing. The index i is called the mode of A. Give an $O(\log n)$ algorithm that finds the mode of A.	4
1(b)	Assume that you are given an array $A[1 \ldots n]$ of distinct numbers. Give an algorithm to reverse the elements of this array.	2
1(c)	Prove that the average case complexity of Bubble Sort is $O(n^2)$, when we are sorting an array of size n.	4
2(a)	Establish the correctness / incorrectness of the statements: (i) $10n^3 + 15n^4 + 100n^2 2^n = O(n^2 2^n)$, (ii) $3^n = O(2^n)$.	2
2(b)	Find the Big-O complexities of the functions in Fig. 1 and Fig. 2.	2+2
2(c)	Following functions is given. Write an equivalent mathematical function $f(n)$ for this function, and then find the Big-O complexity of this function. Use the definition of Big-O and mention the values of c and n_0 explicitly.	4
3(a)	What is the policy, followed in Stack implementation? Stack is a linear or non-linear data structure.	2
3(b)	Give two situations/applications where Stack data structure will be preferred to use.	2
3(c)	We want to implement *push* and *pop* operations on Stack with the help of a linked list. Give algorithms for these operations; taking *start* as the starting address of list, *start* → *info* as the data part of *node* and *start* → *next* as the pointer to next *node*.	4
3(d)	What are the complexities of these operations in your algorithms?	2

```
void Function (int n) {
int i = 1, s = 1;
while (s <= n) {
i++; s = s+i;
printf("!"); }
} //Fig. 1//
```

```
void Function (int n) {
for (int i=1; i<=n; i++)
for (int j=1; j<=n; j=j+i)
printf("!");
}
//Fig. 2//
```

```
void Function (int n) { if (n <= 1) return;
for (int i=1; i<=3; i++) Function(n - 1); }
```