Data Structures and Algorithms - Short Answers

Name: Rahul Goel

Batch- CSE-O2

Reg No: RA1911030010094

1) Insertion sort is a simple comparison based sorting algorithm. It inserts every array element into its proper position unlike bubble sort which repeatedly compares and swaps if needed adjacent elements in every pass. Even though the time complexity is same in the best case scenario (N swaps) insertion sort is faster than bubble sort due to its comparing based algorithm.

2)
```
#include<bits/stdc++.h>
using namespace std;
 int main()
 {
int size;
int array[100];
int array2[100]={0};
cin>>size;
for(int i=0;i<size;i++)
 {
  cin>>array[i];
 }
for (int i = 0; i < size; i++)
 {
   int count= array[i];
   array2[count]= array2[count]+1;
}
int big= array2[0];
int pos =0;
for(int i=0;i<size;i++)
{
   if(array2[i]>big)
    {
      big= array2[i];
      pos= i;
    }
```

```
}
cout<<pos<<" "<<big;


return 0;
}
```

3) The complexity of the given program is O(n).

An algorithm is said to take linear time, or O(n) time, if its time complexity is O(n). This means that the running time increases at most linearly with the size of the input.

This means that there is a constant K such that the running time is at most Kn for every input of size n.

Since the given function runs from i=1 to i<=n its time complexity grows linearly.

The run time solely depends on n .

Therefore the time complexity in O(n).

4) To initialise 1D arrays using pointers :

(i) – Declare the integer array. Eg int arr[] = { 10, 20 , 30 , 40}

(ii) Declare the pointer (integer type) – int*p

(iii) Initialise the pointer to make it point to the base address of the array. p = a.

To access the elements
(i) Use the *(p+n) formula where n is the number of elements

Eg: *(p+0) returns a[0]

*(p + 1) returns a[1]

*(p+2) returns a[2] and so on

5) #include<stdio.h>

```
    struct student{
        char name[40];
        int rno;
        struct subject{
                int s1, s2, s3;
                float avg;
        }sub;
   }st, *sptr = &st;
   int main{
   printf("Name : ");
   scanf("%s", (*sptr).name);
   printf("Rno : ");
   scanf("%d", (*sptr).rno);


   printf("\nsub1 marks : ");
   scanf("%d",&sptr->subject.s1);

   printf("\nsub2 marks : ");
   scanf("%d",&sptr->subject.s2);

  printf("\nsub3 marks : ");
  scanf("%d",&sptr->subject.s3);

  total = (sptr->subject.s1 + sptr->subject.s2 +
  sptr->subject.s3);
  return 0;
}
```