Tutorial 2 & Practical 02

Q1 Implement the following linear search algorithm in C and call it within main() with appropriate data.

```
found =0;
position = -1;
index =0;
while (index < size  && !found) {
      if(data[index]==key)
                {    found = 1;
                     position = index;
                     break;    }
      index++;
}
return position;
```

Estimate the complexity of the above algorithm, giving reasons, in terms of Big-O.

Q2 Implement the following linear search algorithm in C and call it within main() with appropriate data.

```
for(ind=0; ind<size; ind++)
        if(data[ind]==key) break;
return ind;
```

Estimate the complexity of the above algorithm, giving reasons, in terms of Big-O.

03. Implement the following Binary search algorithm in C and call it within main() with appropriate data.

```
first = 0;
last = size -1;
found = 0;
Position = -1;
While(!found && first <=last) {
middle = (first+last)/2;
if(array[middle] == key)  {   found = 1;
               position =middle;   }
```

```
else if(key<array[middle]) last = middle - 1;
     else first = middle + 1;
 }
return position;
```

Estimate the complexity of the above algorithm, giving reasons, in terms of Big-O.

Q4 Code the following Binary search algorithm in C and call it within main() with appropriate data.

```
1    intbsearch(float key, float data[], int size) {
2         int mid= size/2;
3         if(key==data[mid]) return mid;
4         if(size==1) return -100;
5         if(key<data[mid]) return bsearch(key, data, mid);
6         if(key>data[mid])
7         return mid+1+bsearch(key, &data[mid+1], size-1-mid);
8    }
```

Carryout a Desk-Check for the above algorithm for the following data set:

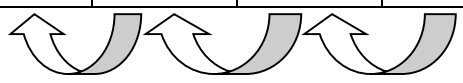
data[] = { 12, 16, 23, 28, 34, 42, 47, 55, 64, 65, 66, 72 }

key = 54

Estimate the complexity of the above algorithm, giving reasons, in terms of Big-O.

Desk Check for Q4 : This is bit different what we have done at the class so focus on the code

Var	**1st call**	**2nd call**	**3rd call**	**4th call**
key	54			
data	&data[0]			
size	12			
mid	6			
data[mid]	47			
return	7 + ?			



The 1st function call returns 7+(-100) = -93 i.e Item not found

note:

When Item is not in the array we need to return an invalid index.

Since the last line (line 7) in bsearch returns (mid+1+ return value of the next call), in line 4, we need to return a large negative value so that after adding all (mid+1) values in successive function calls/returns, still the final return value is evaluated to be a negative integer (an invalid index).