8-) If $n<10$ its best case $T_{Best}(n)=O(1)$ because in $n<10$ there no loop only constant complexity statements and return.

In worst and Average case it will $T_{worst}(n)=T_{Average}(n)=O(\log n)$

because outter loop has constant complexity $O(1)$ and in while loop n decreased by division ($n=n/i$) so its logarithmic.

```
for (i=9; i>1; i--) —— O(1) constant loop
    while (n%i==0)
        n=n/i; —— logarithmic decrease by division
```

Part 2

1-)

```
func (Points arr, int arr_len, Points findPoint)
{
    min = sqrt (pow(arr[0].x - findPoint.x, 2) + pow(arr[0].y - findPoint.y, 2))
O(arr_len) —— for(int i=0; i<arr_len; i++)
    {
        temp = sqrt(pow(arr[i].x - FindPoint.x, 2) + pow(arr[i].y - findPoint.y, 2))
        if(temp < min)
            min = temp
    }
    return temp;
}
```

$T_{best}(n)=T_{worst}(n)=T_{ave}(n)=O(n)=\Omega(n)=\Theta(n)$

2-a)

```
PartA (int arr, arr_len) {
    for(i=1; i<arr_len-1; i++) {
        if(arr[i] <= arr[i+1] && arr[i] <= arr[i-1])
            return arr[i]
    }
}
```

$T_{Best}(n)=O(1)$ second element can be local min so in this case for loop only iterate one time.

$T_{worst}(n)=T_{Average}(n)=O(n)$ There can be no local min in array