NAME :- SADEEM SATTAR
ID :- 19K-1102
BSE - 5A

ASSIGNMENT No 2 :-

Date

QUESTION No 1 :

a) int Find Duplicate (int array , size)
{ for (int i=0 ; i< size ; i++) O(n)
for (int j=0 ; j< size ; j++) O(n)
{ if (array [i] == array [j] && i != j) $O(n^2)$
return array [i];
}
return -1 ; indicate no duplicate element found
}. $O(n^2)$.

b) • First Find maximum number of an array.
• Create dupplicate array of size = maximum number.

int find Duplicate (array , size)
int max = max_num (array , size);
int count [max];
for (int i=0 ; i< max ; i++), O(n).
count [i] = 0;
int i , flag = 0 , j = 0 ;
for (i= 0 ; i< size ; i++) O(n)
count [array[i]]++ ;
if (count [array [i]] > 1)
flag=1; Break; j= i ; break;
if (flag == 0)
return -1 if not found
else
return ~~count~~ array [j] ; if found return that element

O(n)

Date

QUESTION No 2:-

a) • First sort array using merge sort in (n log n)

• Now to apply zig-Zag attribute use.

```
int j=0; temp;
For (i=0 ; i < size ; i = i+2)          ⟶  O(n)
{   j=i;
    temp = array [i];
    array [i] = array [++j];
    array [j] = temp;
}
```

$O(n \log n)$.

b)

```
void Arrange (array, size).
int flag = 0;
for (int i=0; i<size+1; i++)          — O(n)
{ if (flag==0)
    { if (array[i] < array [i+1])
        swap (array [i], array [i+1]).
    }
  else
  { if (array[i] > array [i+1]).
      swap (array [i], array [i+1]).
  }
  flag = 1;
}
```

$O(n)$.

Scanned with CamScanner

QUESTION NO 2:-

a) • First sort array using merge sort in (n log n)

• Now to apply zig-Zag attribute use.

```
~~void~~   int j=0, temp;
    For (i=0 ; i < size ; i=i+2)          ⟶    O(n)
    {   j=i;
        temp = array [i];
        array [i] = array [++j];
        array [j] = temp;
    }.
```

O(n logn).

b)
```
void Arrange (array, size).
    int flag= 0;.
    for (int i=0; i<size ~~+1~~ ; i++).      — O(n)
    { if (flag ==0)
            { if (array [i] < array [i+1]).
                swap (array [i], array [i+1]).
            }.
        else
        { if (array[i]> array [i+1]).
            Swap (array [i] , array [i+1]).
        }
        flag = 1;
    }.
```

O(n).

Scanned with CamScanner

Date

QUESTION No 3:-

→ int calculate_square (int num).
int res=0
{ for (int i=0 ; i< num; i++) → O(n)

res = res+ num

return res;

} O(n).

→ int calculate-square (int num).
{ if (num = =0) return 0;
if (num < 0) num = - num;
int divide = num >>1 ;
if (num & 1)
return ((~~sqr~~ calculate-square(divide)*2) +
(divide * 2) +1);

else
return (calculate-square (divide) <<2);

} O(logn):

Scanned with CamScanner

Date

QUESTION NO 4:

→ int Max_SubArray_Sum (array, size)

```
int  max sum = 0 , sum = 0;
for (int i=0; i<size ; i++).
{        sum+= array[i].
         if (maxsum < sum)
             maxsum = sum.
         if (sum <0)
             sum=0.
}. return  max sum;.
```

→ QUESTION NO 5 :-

Binary Search   O(log n).

a)
```
int  Binary Search (array, L, r, key).
{  if (L<r)
       mid = (L+r)/2 ;
       if  array[mid] == key
           return mid;.
       else if (array[mid] > key)
            return  BinarySearch (array, L, mid-1 , key);
       else if (array[mid] < key
            return  Binarysearch (array, mid+1, r, key);
}.
```

suppose Array of 11 element :-

	0	1	2	3	4	5	6	7	8	9	10
int array[11] = {	2,	4,	5,	6,	7,	$\boxed{8}$	12,	13,	14,	18,	20 }.

key = 18

- L=0 ; r=11 ; mid = 0+11/2 = 5 ; array[5] < key.; 8< 18.

	5	6	7	8	9	10
→	8	12	13	$\boxed{14}$	18	20

Scanned with CamScanner

Date

- L = 6 ; r = 11 , mid = 6+11/2 = 8 ; array [8] < key ; 14<18.

→ 14 18 20

- L 9 ; r = 11 ; mid = 10 array[10] > key 20>18. → 14 18 $\boxed{20}$
- L = 9 ; r = 10 ; mid = 9 array[9] = key 18 = 18. → $\boxed{18}$ 20.

b) jump Search :- $O(\sqrt{n})$

```
int   jump-search (int a[], int size, int key)
{   int   i = 0;
    int step = sqrt (size);
    while (step< size  &&  a[step] <= key)
    {   i = step;          → store prev value of step.
        step = step+ sqrt (size);
        if (step > size-1)
            return -1;
    }

    for (int x = i ; x< step ; x++)
     {   if (a [x] == key)
            return x.
     }
        return -1;
 }.
```

11 = array size.

(1) int a[11] = { 2, 4, 5, 6, 7, 8, 12, 13, 14, 18, 20 }. sqrt (11) = 3.

step = 3

2 int a[11] = { 2, 4, 5, 6, 7, 8, 12, 13, 14, 18, 20 }. step = step + sqrt(11)

Step 6 :-

int a [11] = { 2, 4, 5, 6, 7, 8, 12, 13, 14, $\boxed{18}$ 20 }. → step 9 :-

18 = key.

Date

→ Pros And Cons :-

(i). Binary search → $O(\log n)$
jump Search → $O(\sqrt{n})$

(2) In jump search we traverse the array exactly once no resursive call.
But in binary search. we traverse ~~the~~ back array which make it $O(\log n)$ every time.

b) Interpolation Search :-
Interpolation search is an improvement of Binary search.
It searches the element in range which has higher probability of ~~getting~~ having the key element.

```
int interpolation-search (a[], size, key).
{ int start, end, pos;
  start = 0;
  end = size - 1;
  while (start <= end && key >= a[start] && key <= a[end])
  { pos = start + (double(end-start) / a[end] - a[start]) *
          (key - a[start]));

      if (a[pos] == key)
        return pos;
      if (a[pos] < key)
          start = pos + 1;
      else
          end = pos - 1;
  }
  return -1; }.
```


Product of SAKEENA
Teacher's Signature

Scanned with CamScanner

Date

→ Exponential Search:-

Exponential search is based on binary search. Find element on the basis of range and has two stages. First is to find the range and second is to find the element in the range using binary search.

QUESTION No 6:-

Heap Sort:-

```
↦ void heap-sort (int a[], int size)
{ for (int i = (size/2) - 1 ; i >= 0 ; i--)
        heap-adjust (a, size, i);

    for (int i = size - 1 ; i >= 0 ; i--)
    {       int swap = a[0]
                a[0] = a[i].
                a[i] = swap.
            heap-adjust (a, i, 0);
    }
}.

↦ void heap-adjust (int a[], int size, int i)
{   int largest = i;
    int left = 2*i + 1;
    int right = 2* i + 2;

    if (left < size && a[left] > a[largest]).
        largest = left;
```

Scanned with CamScanner

Date

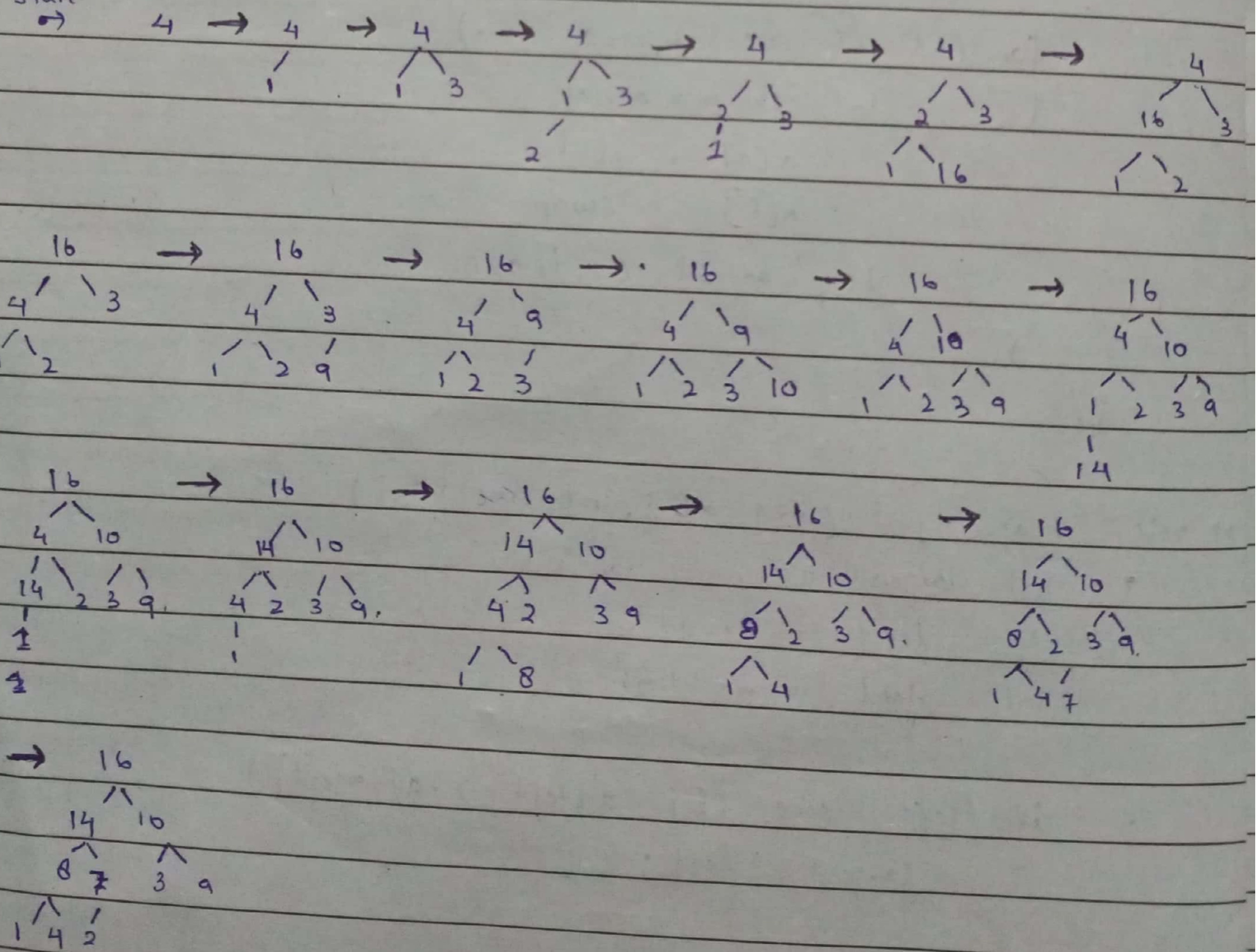
```
if (right <size  && a[right] > a[largest] )
    largest = right;

if (largest != i)
{
    int swap = a[i];
    a[i] = a[largest];
    a[largest] = swap;
    heap-adjust (a, size, largest);
}
};
```

Apply heap Sort :-

4, 1, 3, 2, 16, 9, 10, 14, 8, 7.

start

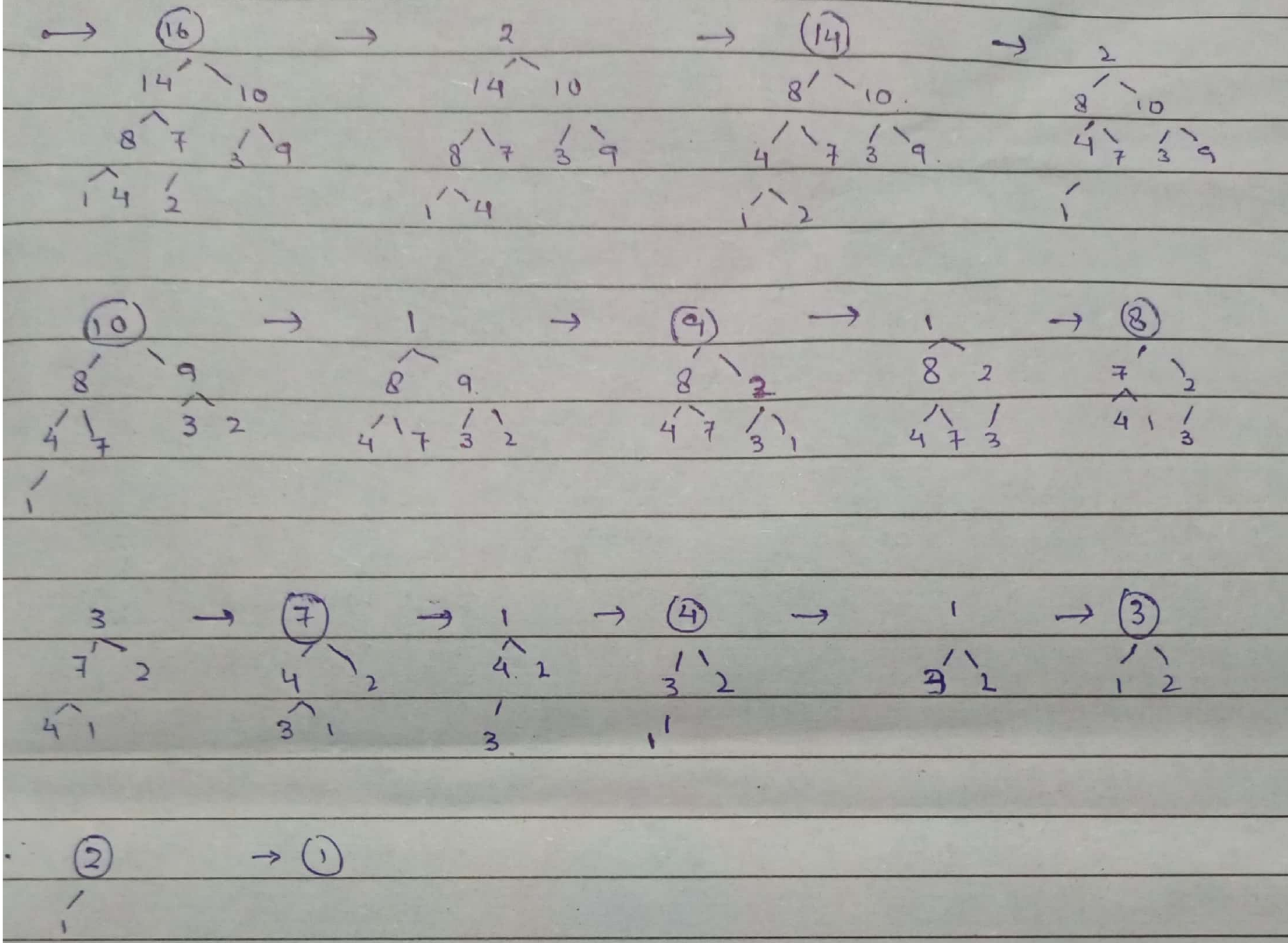


Page No.

Product of

Scanned with CamScanner

Date



Sorted Array.

→ 1, 2, 3, 4, 7, 8, 9, 10, 14, 16