Assignment #1

Date

Name: Hameez Ahmed Siddiqui
Roll No: 20k-0242
Section: BCS-5B
Instructor: Sir Farrukh Salim

X

Answer 1

Brute Force Algorithm:-

```
int main() {
int CurrSum
    int arr[] = {1,2,-1,2,-4,6,-1,12,-10,1,2,-1};
    int CurrSum = arr[0];
    int maxSum = CurrSum;

for(int i=0; i<12 ; i++)  ——— O(n)
{
    CurrSum = arr[i];
    if( CurrSum > maxSum )
        maxSum = CurrSum;
for (int j=i+1; j<12 ; j++ )  —— O((n+1)n)
    {
        CurrSum += arr[j];
        if( CurrSum > maxSum )
            maxSum = CurrSum;
    }
}
}
```

Time Complexity
$O(n^2)$

Scanned with CamScanner

Date

Algorithm in O(n) :-

```
int main()
{
    int arr[] = {1, 2, -1, 2, -4, 6, -1, 12, -10, 1, 2, -1};
    int CurrSum = 0;
    int MaxSum = 0;

    for(i=0; i<12; i++)        — T(O(n))
    {
        ~~CurrSum =~~
        if (CurrSum + arr[i] > arr[i])
            CurrSum = CurrSum + arr[i];
        else if (CurrSum + arr[i] < arr[i])
            CurrSum = arr[i];
        else if (CurrSum + arr[i] < 0)
            CurrSum = 0;

        if (CurrSum < ~~Global~~ MaxSum)
            MaxSum = CurrSum;
    }

}
```

Time Complexity is O(n)

RC

Scanned with CamScanner

Date

Writing Code for O(1):

Answer2

2.a)

```
A = [3,1,2,4]
B = [5,1,6,2]
X = 10

b = 0
A.Sort() // using divide and Conquer
         Sorting technique (nlogn)

while (b< len(b)
{
   temp = X - B[i]
    a = binary_search(A, temp, len(A))
    if a != -1
      return a, b
      a++=1
   else
    return N/A
```


Scanned with CamScanner

Date

2b) for i ← 0 to A.length :
add A[i] to a vector P

for i ← 0 to B.length
j = linearSearch in vector P(X - B[i])
if (j != -1)
return B[i], A[j]
else ~~return~~
return (Not possible)

O(n)

Scanned with CamScanner

Date

Answer3

```
int BinarySearch(int arr[], int l, int r)
{
    if (r >= l)
    {
        int mid = (l+r)/2;
        if (mid == arr[mid])
            return mid;
        if (mid > arr[mid])
        return binarySearch(arr, (mid+1), r);
        else
        return binarySearch(arr, l, (mid-1));
    }

    return -1;
}
```

RC

Scanned with CamScanner

Date

Answer 4

Writing Code for O(n);

```
int main()
{
int arr[10]={1,2,0,5,3,4,0,0,1,10};

int c=0
for(int i=0;i<10;i++)
  {
   if(arr[i]!=0)
     arr[c++] = arr[i];
  }

for(int j=c ;j<10 ;j++ )
   arr[j]=0;

}
```

Time Complexity : O(n)

Space Complexity: O(1)

Scanned with CamScanner

Date

Answer 5

Answer 5.a)

Array = {0, 2, 4, 6, 8, 10, 12, 14, 16, 19}

key = 19

Binary Search:-

arr[mid] = (l+r)/2 = 10

Since key > 9 check 2nd division of array.

l			r
12	14	16	19

arr[mid] = (l+r)/2 = 14 again check 2nd division of array.

l	r
15	18

arr[mid] = (l+r)/2 = 15 again check 2nd division

18

key founded.

Scanned with CamScanner

Date

~~Using~~ Jump Search:-

mid = $\sqrt{10} = 3.1 \simeq 3$

[0] 2, [4] 6, [8] 10, [12] 14, [16] 19

→ linear Search since 19 > 16.

not found

key found [19]

Binary Search is a faster and efficient algorithm that works on divide and Conquer principle Binary Search can only be used when data is sorted.

Jump Search is less costly then ~~is~~ Binary Search as it requires less Comparisions then linear Search. Meanwhile JumpSearch only works on Sorted Arrays and less efficient then binary Search.

Scanned with CamScanner

Date

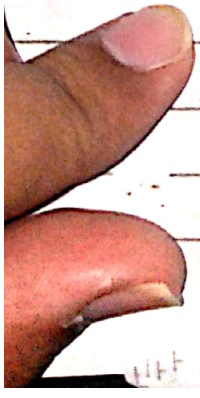
Answer 5b

Interpolation Search :-

It is an improvement of binary search as it is applicable on array with uniformly distributed values. Here we search for element having closest value, then we check if the value is smaller or greater then key and we discard the portion not containing the key, then again search for the position using low and high values.

Exponential Search.

It has same procedure as jump search except we multiply i with 2 if key not found hence doubling the amount jump at each iteration till arr[i] > key, then we apply binary search.

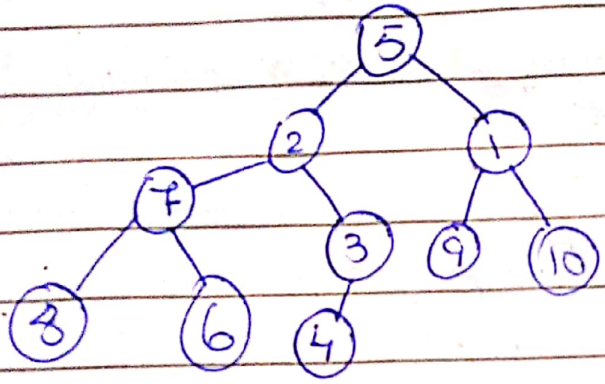


Scanned with CamScanner

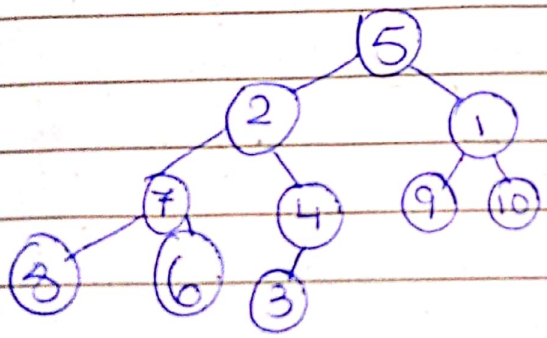
Date

Answer 6

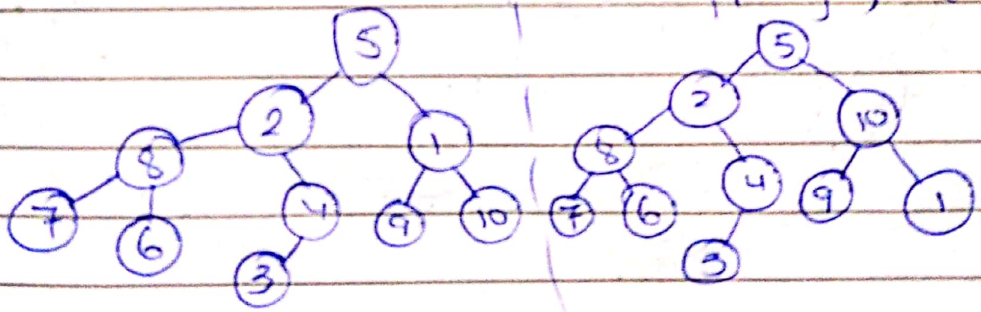
[5, 2, 1, 7, 3, 9, 10, 8, 6, 4]



Swapping 3, 4



Swapping 7, 8

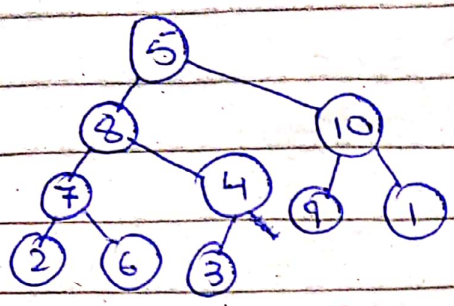


Swapping 1, 10, 9

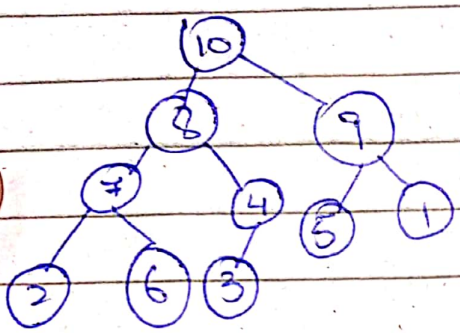
RG

Scanned with CamScanner

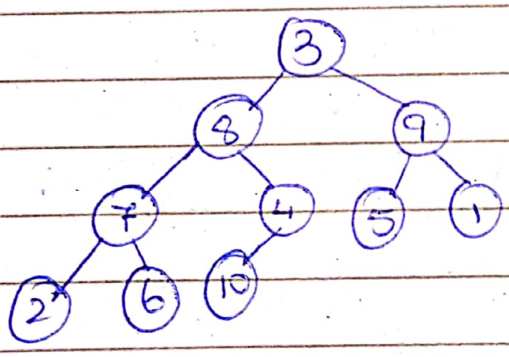
Swapping 2,8 & 2,7



Swapping 5,10 & 5,9



Swapping 10, 3

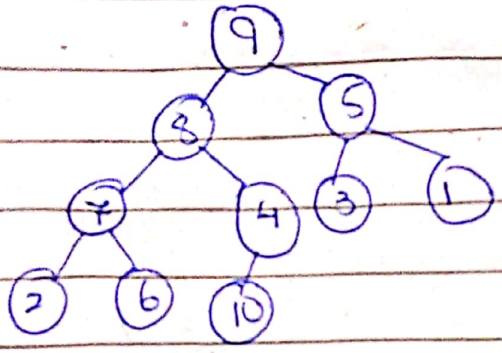


Scanned with CamScanner

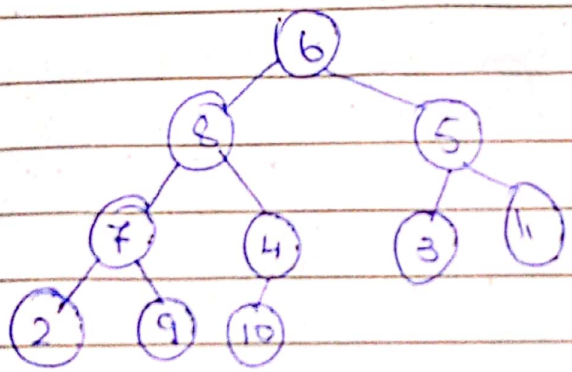
Date

Dao Sorting Using heapsort

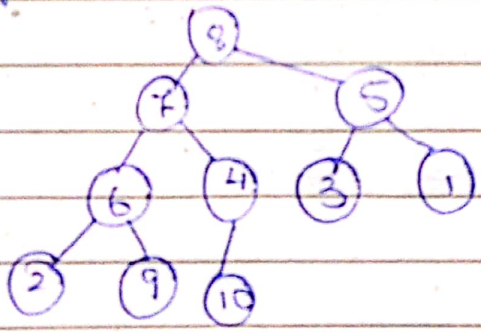
heapify on 3



Swap (9, 6)

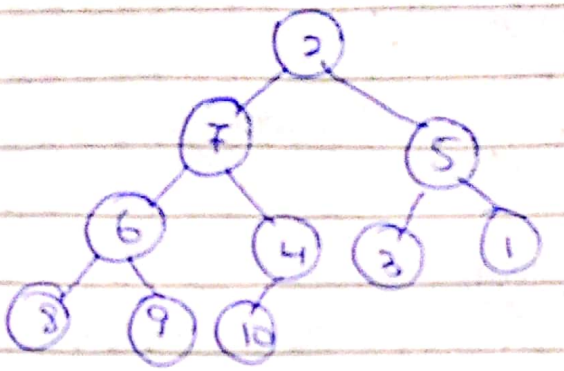


Swap (8, 2)

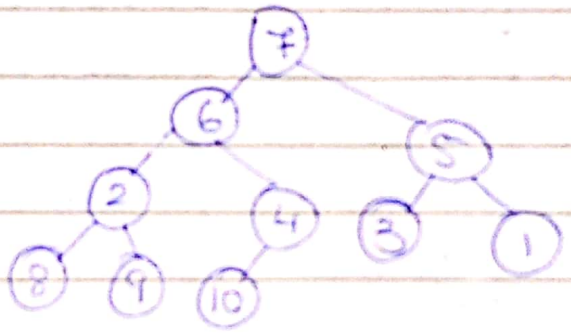


Scanned with CamScanner

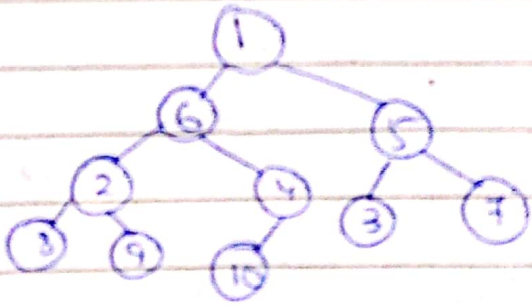
Swap (2,7) & (2,6)



Swap (7,1).



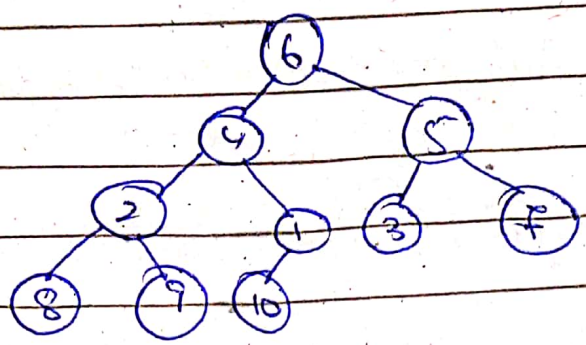
Heapifying 1



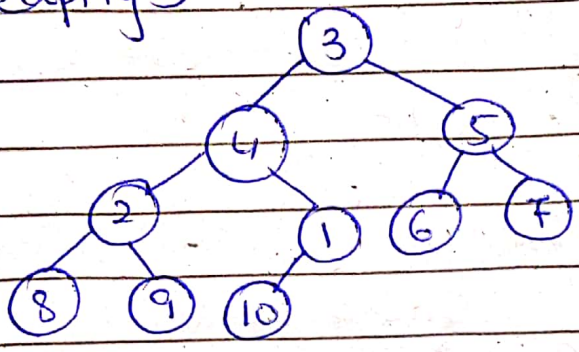
Scanned with CamScanner

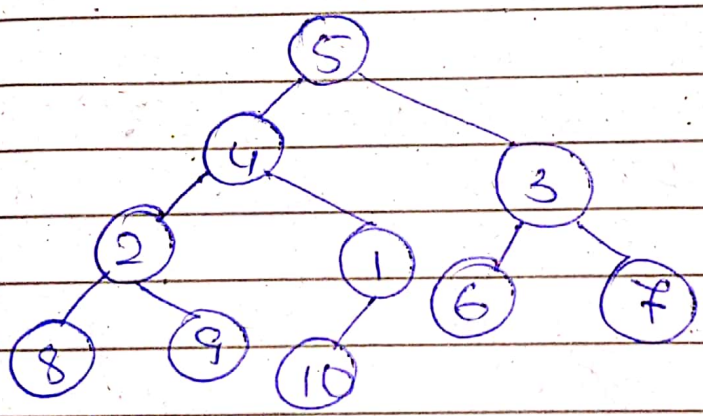
Date

Swap (6,3)

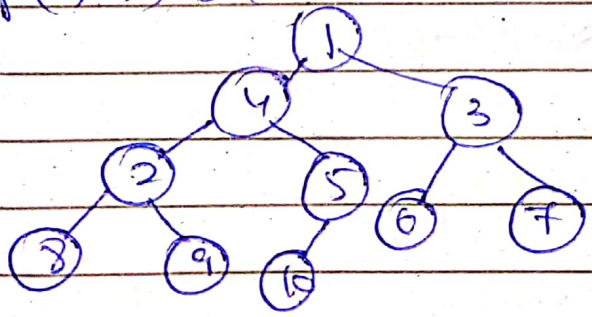


heapify 3





Swap(1,4) & (1,2)

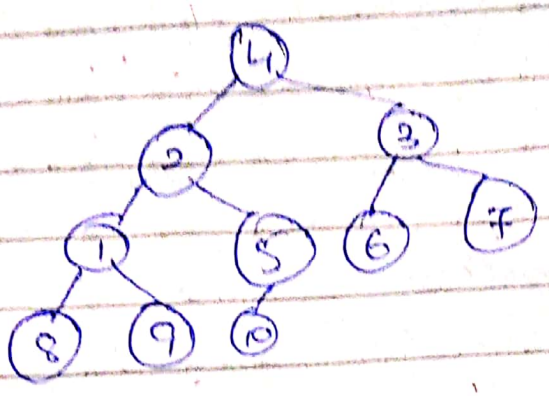


RG

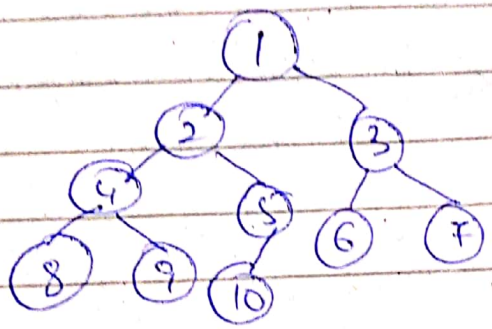
Scanned with CamScanner

Date

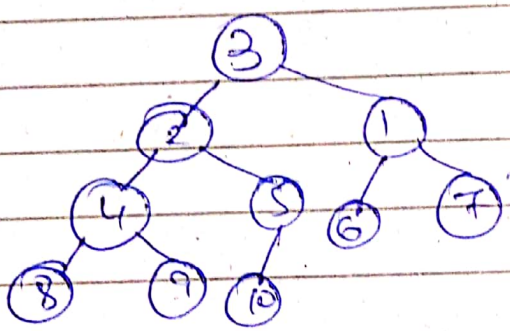
Swap (4, 1)



Heapify 1



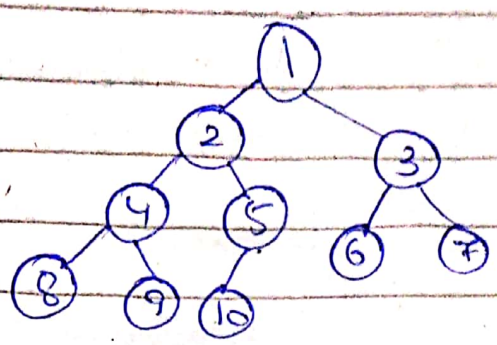
Swap (1, 3)



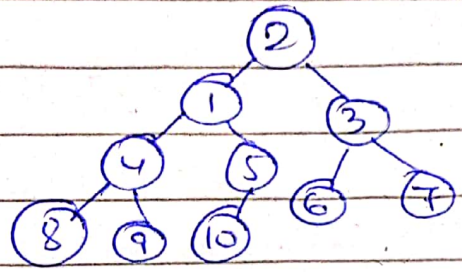
Scanned with CamScanner

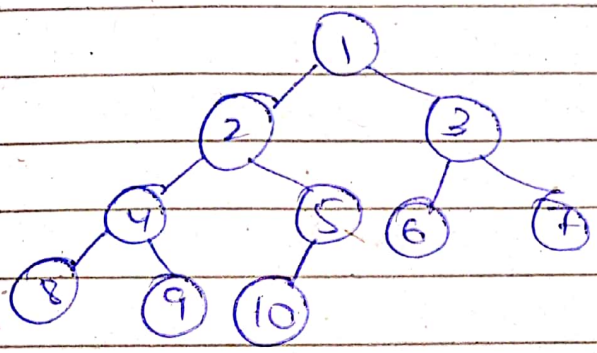
Date

heapify 1



Swap (0, 1)





Sorted Array = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Scanned with CamScanner