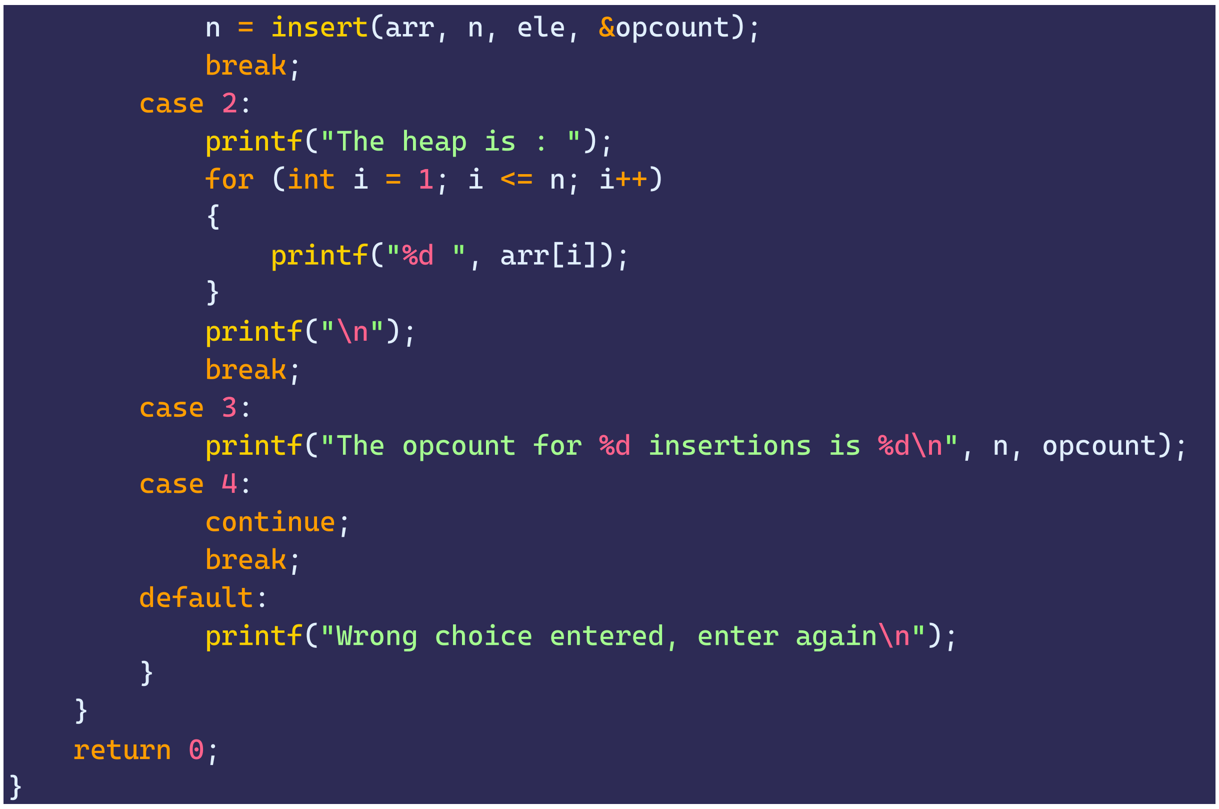
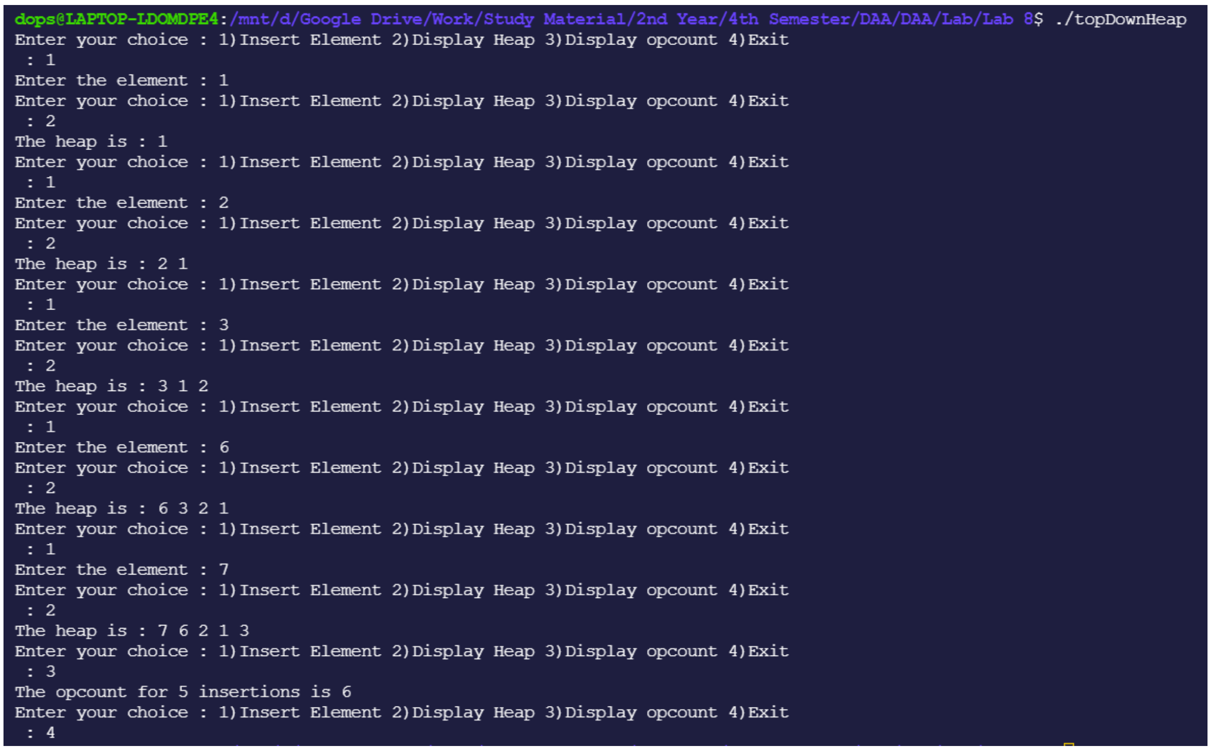
Lab 8 : Dipesh Singh - 190905520

Question 1 : Write a program to create a heap for the list of integers using top-down heap construction algorithm and analyze its time efficiency. Obtain the experimental results for order of growth and plot the result.

```
#include <stdio.h>
#include <stdlib.h>

void heapify(int arr[], int n, int *op)
{
    if (n <= 1)
    {
        return;
    }
    if (arr[n] > arr[n / 2])
    {
        (*op)++;
        int temp = arr[n];
        arr[n] = arr[n / 2];
        arr[n / 2] = temp;
    }
    heapify(arr, n / 2, op);
}

int insert(int arr[], int n, int val, int *op)
{
    n = n + 1;
    arr[n] = val;
    heapify(arr, n, op);
    return n;
}

int main()
{
    int arr[100];
    int choice = 0, ele;
    int n = 0;
    int opcount = 0;
    while (choice < 4)
    {
        printf("Enter your choice : 1)Insert Element 2)Display Heap 3)Dis
play opcount 4)Exit\n : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element : ");
            scanf("%d", &ele);
```

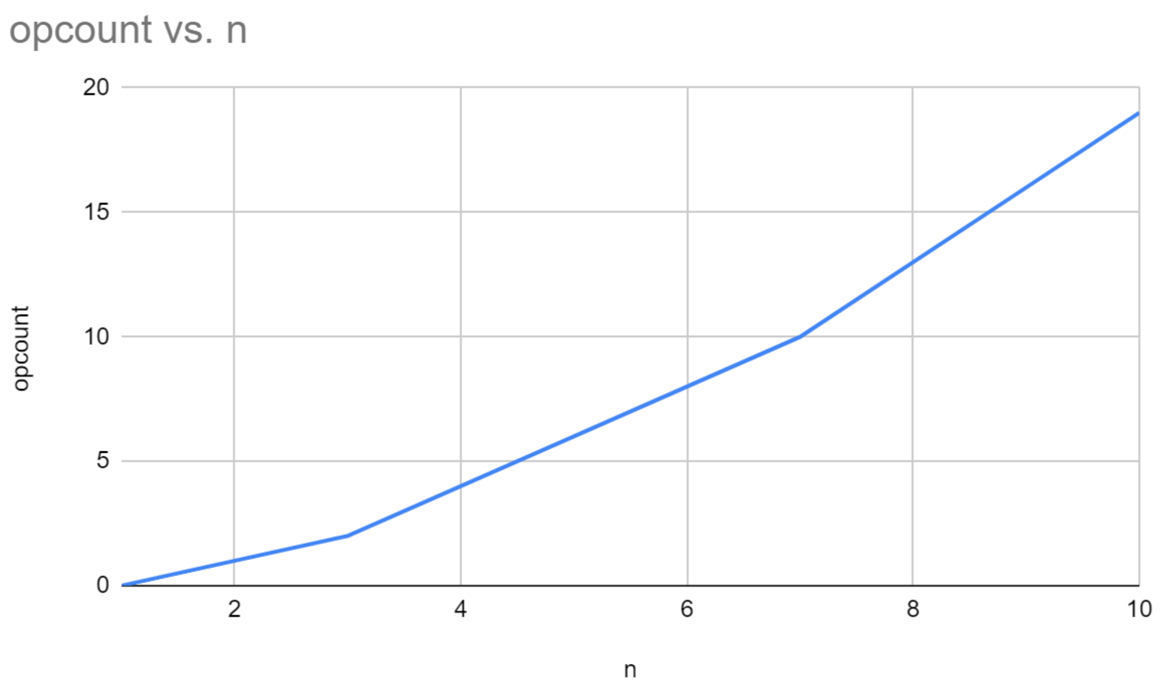


```c
            n = insert(arr, n, ele, &opcount);
            break;
        case 2:
            printf("The heap is : ");
            for (int i = 1; i <= n; i++)
            {
                printf("%d ", arr[i]);
            }
            printf("\n");
            break;
        case 3:
            printf("The opcount for %d insertions is %d\n", n, opcount);
        case 4:
            continue;
            break;
        default:
            printf("Wrong choice entered, enter again\n");
        }
    }
    return 0;
}
```



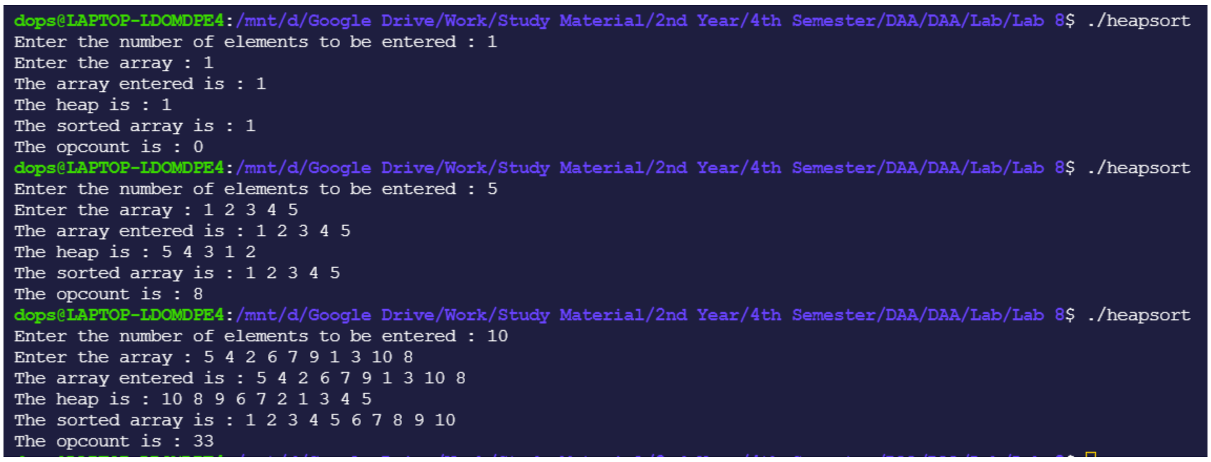
```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 8$ ./topDownHeap
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 1
Enter the element : 1
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 2
The heap is : 1
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 1
Enter the element : 2
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 2
The heap is : 2 1
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 1
Enter the element : 3
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 2
The heap is : 3 1 2
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 1
Enter the element : 6
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 2
The heap is : 6 3 2 1
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 1
Enter the element : 7
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 2
The heap is : 7 6 2 1 3
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 3
The opcount for 5 insertions is 6
Enter your choice : 1)Insert Element 2)Display Heap 3)Display opcount 4)Exit
 : 4
```

n	opcount
1	0
2	1
3	2
4	4
5	6
6	8
7	10
8	13
9	16
10	19

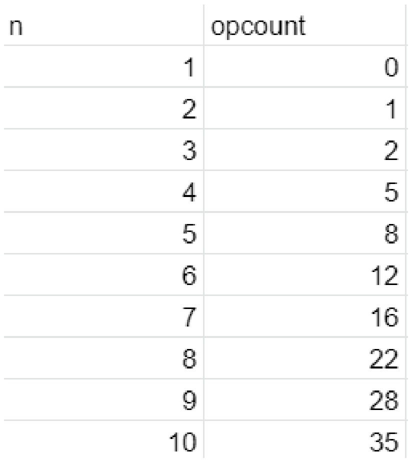


Question 2 : Write a program to sort the list of integers using heap sort with bottom up max heap construction and analyze its time efficiency. Prove experimentally that the worst case time complexity is O(n log n)

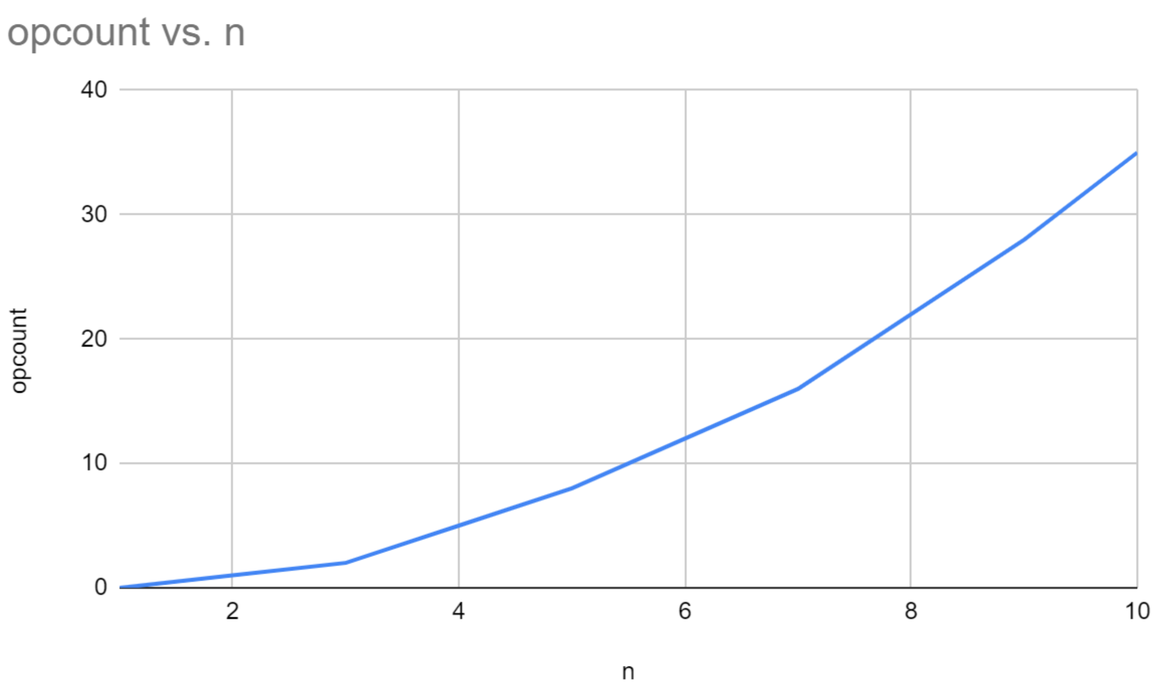
```c
#include <stdio.h>
#include <stdlib.h>

void heapify(int arr[], int n, int *op)
{
    for (int i = n / 2; i >= 1; i--)
    {
        int v = arr[i];
        int k = i;
        int flag = 0;
        while (!flag && 2 * k <= n)
        {
            (*op)++;
            int j = 2 * k;
            if (j < n)
            {
                if (arr[j] < arr[j + 1])
                {
                    j++;
                }
            }
            if (v < arr[j])
            {
                arr[k] = arr[j];
                k = j;
            }
            else
            {
                flag = 1;
            }
        }
        arr[k] = v;
    }
    return;
}

int maxDel(int arr[], int n, int *op)
{
    int temp = arr[1];
    arr[1] = arr[n];
    arr[n] = temp;
    n = n - 1;
    heapify(arr, n, op);
    return n;
}

int main()
{
    int n;
    printf("Enter the number of elements to be entered : ");
    scanf("%d", &n);
    int arr[n + 1];
    printf("Enter the array : ");
    for (int i = 1; i <= n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("The array entered is : ");
    for (int i = 1; i < n + 1; i++)
    {
        printf("%d ", arr[i]);
    }
    int opcount = 0;
    heapify(arr, n, &opcount);
    printf("\nThe heap is : ");
    for (int i = 1; i < n + 1; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\nThe sorted array is : ");
    int t = n;

    for (int i = 1; i <= t; i++)
    {
        n = maxDel(arr, n, &opcount);
    }
    for (int i = 1; i <= t; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\nThe opcount is : %d\n", opcount);
    return 0;
}
```



```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 8$ ./heapsort
Enter the number of elements to be entered : 1
Enter the array : 1
The array entered is : 1
The heap is : 1
The sorted array is : 1
The opcount is : 0
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 8$ ./heapsort
Enter the number of elements to be entered : 5
Enter the array : 1 2 3 4 5
The array entered is : 1 2 3 4 5
The heap is : 5 4 3 1 2
The sorted array is : 1 2 3 4 5
The opcount is : 8
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 8$ ./heapsort
Enter the number of elements to be entered : 10
Enter the array : 5 4 2 6 7 9 1 3 10 8
The array entered is : 5 4 2 6 7 9 1 3 10 8
The heap is : 10 8 9 6 7 2 1 3 4 5
The sorted array is : 1 2 3 4 5 6 7 8 9 10
The opcount is : 33
```



n	opcount
1	0
2	1
3	2
4	5
5	8
6	12
7	16
8	22
9	28
10	35


opcount vs. n
opcount
n