Name: Pratik Rameshwar Shinde

Roll No.: 2274

Class: SE-IT

Div: B-3

Subject: DSA LAB

Practical 9: Heap Sort

Aim: Implement Heap sort to sort given set of values using max or min heap.

**************************PROGRAM**************************

```
#include <iostream>  using
namespace std;
// A function to heapify the array. void
MaxHeapify(int a[], int i, int n)
{
        int j, temp; temp
        = a[i];
        j = 2*i;

         while (j <= n)
        {
                if (j < n && a[j+1] > a[j]) j =
                j+1;
                // Break if parent value is already greater than child value.
                if (temp > a[j]) break;
                // Switching value with the parent node if temp < a[j]. else if
                (temp <= a[j])
                {
                        a[j/2] = a[j];
                        j = 2*j;
                }
        }
        a[j/2] = temp; return;
}
void Build_MaxHeap(int a[], int n)
{
```

```
            int i;
            for(i = n/2; i >= 1; i--)
                        MaxHeapify(a, i, n);
}
void Max_HeapSort(int a[], int n)
{
            int i, temp;
            for (i = n; i >= 2; i--)
            {
                        // Storing maximum value at the end.
                        temp = a[i]; a[i] = a[1]; a[1] = temp;
                               // Building max heap of remaining element.
                        MaxHeapify(a, 1, i - 1);
            }
}
void min_heapify(int a[],int i,int n)
{
   int j, temp;
temp = a[i];     j =
2 * i;     while (j <=
n)
   {
       if (j < n && a[j+1] < a[j])
           j = j + 1;        if (temp
< a[j])           break;        else
if (temp >= a[j])
       {           a[j/2] =
a[j];           j = 2 * j;
       }
   }
   a[j/2] = temp;
   return;
}
void build_minheap(int a[], int n)
{
   int i;
   for(i = n/2; i >= 1; i--)
   {
        min_heapify(a,i,n);
    }
}
void Min_HeapSort(int a[], int n)
{
            int i, temp; for (i = n; i
            >= 2; i--)
            {
                        // Storing minimum value at the end.
                        temp = a[i]; a[i] = a[1]; a[1] = temp;
                        // Building max heap of remaining element.
                        min_heapify(a, 1, i - 1);
            }
}
void print(int arr[], int n)
{
cout<<"\nSorted Data ";

            for (int i = 1; i <=n; i++) cout<<"->"<<arr[i];
            return;
} int main()
{
            int n, i, ch;
            cout<<"\nEnter the number of data elements to be sorted: "; cin>>n;
            int arr[n];
            for(i = 1; i <=n; i++)
            { cout<<"Enter the element "<<i<<": ";
            cin>>arr[i]; }
            // Building max heap.

do {
cout<<"\n\n1. Heap sort using  max heap"; cout<<"\n2.
Heap sort using  min heap"; cout<<"\n3. Exit";


cout<<"\nEnter your choice: ";
cin>>ch; switch(ch) {
```

```
case 1: Build_MaxHeap(arr, n);          Max_HeapSort(arr,
n);
      print(arr, n);
      break;
case 2: build_minheap(arr, n);          Min_HeapSort(arr,
n);
       print(arr, n);
break;

case 3:return 0;
     default:cout<<"\n Invalid choice !! Please enter your choice again."<<endl;
} }
while(ch!=3);
}
```

```
****************************OUTPUT****************************

[admin@fedora ~]$ g++ hfbdsa9.cpp
[admin@fedora ~]$ ./a.out

Enter the number of data elements to be sorted: 5
Enter the element 1: 12
Enter the element 2: 5
Enter the element 3: 87
Enter the element 4: 100 Enter
the element 5: 36



1.  Heap sort using  max heap
2.  Heap sort using  min heap
3.  Exit
Enter your choice: 1

Sorted Data ->5->12->36->87->100

1.  Heap sort using  max heap
2.  Heap sort using  min heap
3.  Exit
Enter your choice: 2
```

```
Sorted Data ->100->87->36->12->5

1. Heap sort using  max heap
2. Heap sort using  min heap
3. Exit
Enter your choice: 3
[admin@fedora ~]$
```