**PUNE INSTITUTE OF COMPUTER TECHNOLOGY DHANKAWADI, PUNE**

**Data Structures And Algorithms(DSA)**

**Assignment No. 09**

**Title : Heap Sort**

**SE-IT-10**  **ACADEMIC YEAR :- 2020-2021**

**Name :- Diptesh Ravindra Varule Roll No :- 23277**

**Source Code:**

**.cppFile :**

//============================================================================

// Name : heapsort.cpp

// Author : Diptesh Varule

// Version : Updating...

// Copyright : Your copyright notice

// Description : Hello World in C++, Ansi-style

//============================================================================

#include"heapsort.h"

#include <iostream>

using namespace std;

int main() {

heap sort;

sort.input();

cout<<"Before sorting "<<endl;

sort.display();

cout<<"\nHeap list is"<<endl;

sort.build\_heap();

sort.display();

cout<<"\nHeap sort start"<<endl;

sort.heapsort();

cout<<"\nSorted output "<<endl;

sort.display();

return 0;

}

**Header File :**

#ifndef HEAPSORT\_H\_INCLUDED

#define HEAPSORT\_H\_INCLUDED

#include<iostream>

using namespace std;

class heap{

int list[20];

int size;

public:

void input();

void display();

void max\_heapify(int ,int);

void heapsort();

void build\_heap();

};

void heap::build\_heap(){

int i;

for(i=size/2;i>=0;i--){

max\_heapify(i,size);

}

}

void heap::max\_heapify(int i,int li){

int j, temp;

temp=list[i];

j=2\*i;

while(j<li){

if(j<=li && list[j+1] > list[j])

j+=1;

if(temp>=list[j])

break;

else if(temp<=list[j]){

list[j/2]=list[j];

j=2\*j;

}

}

list[j/2]=temp;

return;

}

void heap::heapsort(){

int i,temp;

for(i=size-1;i>=0;i--){

temp=list[i];

list[i]=list[0];

list[0]=temp;

cout<<"\nHeap tree after root deletion"<<endl;

display();

max\_heapify(0,i-1);

cout<<"\nHeap after Reconstruction"<<endl;

display();

}

}

void heap::display(){

for(int i=0;i<size;i++){

cout<<" "<<list[i];

}

}

void heap::input(){

cout<<"Enter number of elements"<<endl;

cin>>size;

for(int i=0;i<size;i++){

cout<<"Element "<<i+1<<endl;

cin>>list[i];

}

}

#endif // HEAPSORT\_H\_INCLUDED

**Output :**

Enter number of elements

5

Element 1

2

Element 2

8

Element 3

5

Element 4

12

Element 5

9

Before sorting

2 8 5 12 9

Heap list is

12 9 5 8 2

Heap sort start

Heap tree after root deletion

2 9 5 8 12

Heap after Reconstruction

9 8 5 2 12

Heap tree after root deletion

2 8 5 9 12

Heap after Reconstruction

8 2 5 9 12

Heap tree after root deletion

5 2 8 9 12

Heap after Reconstruction

5 2 8 9 12

Heap tree after root deletion

2 5 8 9 12

Heap after Reconstruction

2 5 8 9 12

Heap tree after root deletion

2 5 8 9 12

Heap after Reconstruction

2 5 8 9 12

Sorted output

2 5 8 9 12