```
/* Note: To deal with garbage
         collection I assigned 0 as default value,
         so 0 as a value will not be accepted */

#include <bits/stdc++.h>
using namespace std;

int root = 0, L=1;

//Sort vertex values from lower to upper

void upwardHeapify(int arr[], int n, int Lchild, int
Rchild){
    int i=Rchild;
    while(i>=1 && Lchild>1 && Rchild>2){
        if(arr[i/2]>arr[Lchild]){
            swap(arr[i/2], arr[Lchild]);
        }
        else if(arr[Rchild]!=0){
          if(arr[i/2]>arr[Rchild]){
              swap(arr[i/2], arr[Rchild]);
           }
        }
        i=i/2;
        Lchild=i;
        Rchild=Lchild+1;
    }
}
```

```
//Create new vertex and call upwardHeapify function

void Insert(int arr[], int n, int value){
    int Lchild, Rchild;
    if(root==0){
        root = 1;
        for(int i=0; i<n; i++){
            arr[i] = 0;
        }
        arr[root] = value;
    }
    else{
        Lchild = 2*L, Rchild = (2*L)+1;
        if(arr[Lchild]==0){
            arr[Lchild] = value;
            upwardHeapify(arr, n, Lchild, Rchild);
        }
        else{
            arr[Rchild] = value;
            upwardHeapify(arr, n, Lchild, Rchild);
            L++;
        }
    }
}
```

```
//Sort vertex values from upper to lower

void downwardHeapify(int arr[], int n, int c){
    int N=c/2;
    int i=1;
    while(i<=N){
        int Lchild=2*i, Rchild=(2*i)+1;
        if(arr[Lchild]==0 && arr[Rchild]==0){
            break;
        }
        else if(arr[Rchild]!=0){
            if(arr[Lchild]<arr[Rchild] ||
arr[Lchild]==arr[Rchild]){
                if(arr[i]>arr[Lchild]){
                    swap(arr[i],arr[Lchild]);
                }
            }
            else{
                if(arr[i]>arr[Rchild]){
                    swap(arr[i],arr[Rchild]);
                }
            }
        }
        else{
            if(arr[i]>arr[Lchild]){
                swap(arr[i],arr[Lchild]);
            }
        }
        i++;
    }
}
```

```
//Delete vertex and call downwardHeapify

void Delete(int arr[], int n){
    int c=0;
    for(int i=1; i<n; i++){
        if(arr[i]!=0){
            c++;
        }
    }
    if(c!=0){
        if(c==1){
            arr[c] = 0;
        }
        else{
            swap(arr[c], arr[root]);
            arr[c]=0;
        }

        if(c%2!=0){
            L--;
            if(L==0){
                L=1;
                root = 0;
            }
        }
        downwardHeapify(arr, n, c);
    }
}

//Find root value function

void Find(int arr[]){
    cout <<"Root value = "<<arr[root]<<endl;
}
```

```
//Main function

int main(){
int n;
cout <<"Enter size: ";
cin >>n;
n += 2;
int arr[n], i=1;
int choice, value;

while(i<n-1){
    cout <<"1.Insert 2.Delete 3.Find"<<endl;
    cin >>choice;
    switch(choice){
        case 1:
        cin >>value;
        Insert(arr, n, value);
            break;
        case 2:
            Delete(arr, n);
            break;
        case 3:
        Find(arr);
            break;

    }
    i++;
}

return 0;

}
```